Discriminative and Bayesian techniques for hidden Markov model speech recognition systems

by

Darryl William Purnell

Submitted in partial fulfillment of the requirements for the degree of

PHILOSOPHIAE DOCTOR (ENGINEERING)

in the Faculty of Engineering

UNIVERSITY OF PRETORIA

February 2001

Summary

The collection of large speech databases is not a trivial task (if done properly). It is not always possible to collect, segment and annotate large databases for every task or language. It is also often the case that there are imbalances in the databases, as a result of little data being available for a specific subset of individuals. An example of one such imbalance is the fact that there are often more male speakers than female speakers (or *vice-versa*). If there are, for example, far fewer female speakers than male speakers, then the recognizers will tend to work poorly for female speakers (as compared to performance for male speakers).

This thesis focuses on using Bayesian and discriminative training algorithms to improve continuous speech recognition systems in scenarios where there is a limited amount of training data available. The research reported in this thesis can be divided into three categories:

- Overspecialization is characterized by good recognition performance for the data used during training, but poor recognition performance for independent testing data. This is a problem when too little data is available for training purposes. Methods of reducing overspecialization in the minimum classification error algorithm are therefore investigated.
- Development of new Bayesian and discriminative adaptation/training techniques that can be used in situations where there is a small amount of data available.

One example here is the situation where an imbalance in terms of numbers of male and female speakers exists and these techniques can be used to improve recognition performance for female speakers, while not decreasing recognition performance for the male speakers.

- Bayesian learning, where Bayesian training is used to improve recognition performance in situations where one can only use the limited training data available. These methods are extremely computationally expensive, but are justified by the improved recognition rates for certain tasks. This is, to the author's knowledge, the first time that Bayesian learning using Markov chain Monte Carlo methods have been used in hidden Markov model speech recognition.

The algorithms proposed and reviewed are tested using three different datasets (TIMIT, TIDIGITS and SUNSpeech), with the tasks being connected digit recognition and continuous speech recognition. Results indicate that the proposed algorithms improve recognition performance significantly for situations where little training data is available.

Keywords: speech recognition, hidden Markov model training, minimum classification error, Bayesian adaptation, Bayesian learning, maximum a posteriori parameter estimation, sparse data

Samevatting

Die versameling van groot spraakdatabasisse is nie 'n maklike taak nie. Dit is nie altyd moontlik om groot databasise to versamel, in segmente te verdeel en te annoteer vir elke taak of taal nie. Dit is ook dikwels die geval dat daar 'n wanbalans bestaan in 'n databasis, as gevolg van die onverkrygbaarheid van data vir 'n spesifieke subgroep van individue. Een voorbeeld van so 'n wanbalans is die feit dat daar dikwels meer manssprekers as vrouesprekers is (of andersom). In so 'n geval, sal die herkenner gewoonlik nie goed werk vir vrouens nie, maar sal relatief goed werk vir mans.

Hierdie tesis fokus op die gebruik van Bayes en diskriminerende afrigtingstegnieke om kontinuspraakherkenningstelsels te verbeter in scenarios waar min afrigdata beskikbaar is. Die navorsing waaroor hier gerapporteer word, kan in drie dele verdeel word:

- Oor-spesialisasie word gekarakteriseer deur goeie herkenning vir die afrigdata, maar slegte herkenning vir onafhanklike toetsdata. Hierdie probleem onstaan wanneer te min data beskikbaar is vir afrigtingsdoeleindes. Metodes om oor-spesialisasie in minimum klassifikasiefout afrigting te verminder word dus hier ondersoek.

- Nuwe Bayes en diskriminerende afrigtings- en aanpassingstegnieke om herkenning te verbeter in situasies waar min data beskikbaar is. Een voorbeeld is die situasie waar 'n wanbalans in terme van die aantal vroue- en manssprekers bestaan. Hierdie tegnieke kan gebruik work om die herkenningstempo te verbeter vir die

vrouesprekers.

- Bayes afrigting word gebruik om herkenning te verbeter in situasies waar min data beskikbaar is, en geen taakspesifieke data nie. Hierde metode is uiters berekeningintensief, maar ek glo dat die verbetering in herkenningstempo dit regverdig. Hierdie is, na ons medewete , die eerste keer dat Markov ketting Monte Carlo gebaseerde Bayesian afrigting in verskuilde Markov model spraakherkenningstelsels gebruik word.

Die voorgestelde algoritmes is getoets met drie verskillende spraakdatabasise: TIMIT, TIDIGITS en SUNSpeech. Die take is kontinusyferherkenning en kontinuspraakherkenning. Resultate toon dat die voorgestelde algoritmes goed werk vir situasies waar min afrigtingsdata beskikbaar is.

Sleutelwoorde: spraakherkenning, verskuilde Markov model afrigting, minimum klassifikasiefout, Bayes aanpasing, Bayes afrigting, maksimum a posteriori parameter skating, min afrigdata

Acknowledgements

First and foremost, I would like to thank Professor Liesbeth Botha for the advice and guidance she gave me during the last three years.

I would also like to thank Christoph Nieuwoudt and Johann Holm, with whom I had many interesting and thought-provoking discussions.

Finally, I would like to thank my family and friends for their support and encouragement throughout.

Contents

Chapter 1

Introduction

Communication through speech is an integral part of our lives. Automatic speech recognition for machines (computers) therefore provides a natural and effective way of communicating with machines. We are, however, far from creating a machine which can understand spoken discourse on any subject, by all speakers and for all conditions. Much research and development of speech recognition systems will be required before this goal is achieved.

Hidden Markov models (HMMs) have been the dominant approach to speech recognition since the 1980s [94, 92]. HMMs are statistical models used to characterize the spectral properties of the frames of a pattern.

A *Markov model* is a system that can be described as being in one of N distinct states at any given time. At regularly spaced, discrete times, the system undergoes a change of state, depending on a set of probabilities associated with each state. This is known as a discrete-time Markov process. *Hidden* Markov models are doubly embedded stochastic processes with an underlying stochastic process that is not directly observable (hidden) but can be observed indirectly through another set of stochastic processes that produce the sequence of observations. The observation is therefore a probabilistic function of

the state. Section 2.1 will describe HMMs and their use in speech recognition in more detail.

Hidden Markov model speech recognition systems typically consist of two main parts, namely

1. *Feature extraction.* Here features, which will be used to recognize the utterance, are extracted from the speech signal. One of the most common features used are the Mel-frequency cepstral coefficients (MFCCs), which can be obtained from the power spectrum of the speech signal. Section 2.1.1 gives a detailed description of the feature extraction process as used in this thesis.

2. *Hidden Markov models.* HMMs are used to represent the temporal nature of the speech signal. A word or phonetic unit is typically represented by a single HMM, with each state in the HMM representing an acoustic unit within the word or phonetic unit. In a continuous digit recognition task, for example, HMMs would be used to represent the digits 0 through 9 and other events such as silence and inter word pauses. In continuous density HMMs, a weighted sum of Gaussian distributions is used to represent the probability of an observation (features extracted) being generated by that state. Section 2.1 presents a detailed description of the HMM system used in this thesis.

The assumption made when using a hidden Markov model (HMM) or any statistical model, is that the process can be characterized as a parametric random process. It is furthermore assumed that these parameters can be determined or estimated in a precise, well-defined manner. HMMs are usually trained using the maximum likelihood (ML) criterion. When creating continuous speech recognition systems, sparse training data is often a problem. This limits the effectiveness of the conventional approaches, such as maximum likelihood parameter estimation.

The collection of large speech databases is not a trivial task (if done properly). It is

not always possible to collect, segment and annotate large databases for every task, language or dialect. The collection of large speech databases is an expensive and time consuming task. It is doubtful whether there will ever be a substitute for sufficient, well recorded and annotated data when creating speech recognition systems. However, given that it is not always possible to collect sufficient data, this thesis focuses on using Bayesian and discriminative training algorithms to improve continuous speech recognition systems in scenarios where there is a limited amount of training data available.

1.1 Adaptation

Adaptation is a process for adjusting seed models or training data (non-task-specific) to create more specialized models using a small amount of task-specific adaptation data. There are many applications of adaptation algorithms, including:

- **Speaker adaptation** Speaker adaptation is a well researched and documented [67, 74, 32, 6, 20, 31, 21] example of adaptation, where little speaker-dependent data is available for creating a speaker-specific model. Using the speaker-independent model or dataset of many speakers, adaptation techniques can be used to create an improved speaker-dependent model. It is impractical to use the algorithms described in this thesis for speaker adaptation due to their computational expensive nature.

- **Gender adaptation** It is well documented [117, 51, 118] that usage of gender dependent models for male and female speakers improves performance. Adaptation can be used to improve the gender dependent performance in situations where there is limited training data available. If, for example, we have a reasonable amount of training data for female speakers but little for male speakers, we could adapt the female model or data using the male training data, thereby improving the recognition performance for male speakers.

- **Language and dialect adaptation** As mentioned, the collection of a comprehensive database for a new language or dialect is a difficult and time consuming procedure. Much of the work in multi-language research has focused on creating speech recognition systems which can recognize speech from multiple languages [47, 10, 29, 25], or bootstrapping of new monolingual models for a new language using existing models [116, 102].

  Alternatively, we can apply adaptation techniques to reduce the amount of language or dialect-specific training data required. Recently, some studies [112, 15, 30, 63, 85] have therefore used adaptation techniques to improve the recognition performance for a new language or dialect, using existing models of other languages or dialects.

Two main families of adaptation schemes have been proposed in the past, namely Bayesian adaptation and transformation-based adaptation procedures.

1.1.1 Bayesian adaptation

The maximum *a-posteriori* (MAP) estimation procedure [67, 43, 45] attempts to find the parameters (θ) which maximize the posterior probability of the parameters given the training data, i.e.

$$\theta_{MAP} = \underset{\theta}{argmax} P(\theta|\mathbf{X}) = \underset{\theta}{argmax} P(\mathbf{X}|\theta)P(\theta) \tag{1.1}$$

where X is the training data. $P(\theta)$ is the prior distribution of the model parameters and expresses any knowledge about the parameters or model prior to any data being observed. $P(\mathbf{X}|\theta)$ is the probability of the observation $\mathbf{X}$ being generated by the model with parameters θ. $P(\mathbf{X})$ has not been included in Eq. (1.1) as it is a normalizing constant and does not affect the mode (θ_{MAP}) of the posterior distribution $P(\theta|\mathbf{X})$.

The MAP framework provides a way of incorporating prior information in the estimation process, which is useful when dealing with problems caused by limited training data. This prior information can be subjective or information obtained from non-task-specific data or models. MAP estimation has sometimes been referred to as Bayesian learning in the speech recognition literature. MAP estimation is an approximate Bayesian learning algorithm and will therefore not be referred to as Bayesian learning in this thesis.

When using MAP estimation for adaptation purposes, the non-task-specific information is encapsulated in the prior distribution. For example, in speaker adaptation, speaker independent data or models will be used to create the prior, along with any other subjective information. MAP starts from the seed model performance and converges asymptotically to task-specific performance as the amount of adaptation data increases. The usage of the MAP algorithm will be investigated in Chapter 4.

The following summarizes the advantages and disadvantages of Bayesian and transformation-based (Section 1.1.2) approaches,

- Bayesian adaptation requires relatively large amounts of adaptation data (compared to transformation-based methods).
- Transformation-based methods are typically much faster in that they require very small amounts of training data.
- Transformation-based adaptation has the advantage that it can be applied either directly to the features, or to the HMM parameters.
- A disadvantage of transformation-based methods is that they tend not to take full advantage of larger amounts of adaptation data.
- Transformation-based adaptation is usually text-dependent, whereas Bayesian adaptation is typically text-independent.

- Bayesian adaptation has good asymptotic properties: performance converges to speaker-dependent performance as the amount of adaptation speech increases.

1.1.2 Transformation-based adaptation

Transformation-based techniques estimate a transformation of seed model parameters, thereby creating the new task-specific model. The transformation is typically linear [33, 69], though non-linear transforms have also been used [1]. When using a linear transformation, we estimate the following transformation,

$$\mathbf{x} = \mathbf{A}\mathbf{y} + \mathbf{b}, \tag{1.2}$$

where $\mathbf{y}$ is the seed model parameter vector, $\mathbf{x}$ is the new adapted model parameters, $\mathbf{A}$ and $\mathbf{b}$ are the transformation matrix and offset vector.

The transformation will typically contain far fewer parameters compared to the model we are transforming. We will therefore be able to estimate a reasonably accurate transformation when very little data is available. Generally, when the adaptation data is limited, transformation-based adaptation can therefore efficiently transform all the HMM parameters using cluster-dependent transformations.

Given a seed HMM model ($\mathbf{y}$) with observation densities of the form

$$P(\mathbf{o}|s_t) = \sum_{i=1}^{M} c_s i \mathcal{N}(\mathbf{o}, \mu_{ig}, \mathbf{\Sigma}_{ig}), \tag{1.3}$$

where g is the index of the Gaussian codebook used by state s_t. The transformation

$$\mathbf{x}_t = \mathbf{A}_g \mathbf{y}_t + \mathbf{b}_g, \tag{1.4}$$

results in a adapted model with observation densities of the form

$$P(\mathbf{o}|s_t) = \sum_{i=1}^{M} c_s i\mathcal{N}(\mathbf{o}, \mathbf{A}_g\mu_{ig} + \mathbf{b}_g, \mathbf{A}_g\Sigma_{ig}\mathbf{A}_g^T). \tag{1.5}$$

Only the parameters $\mathbf{A}_g$, $\mathbf{b}_g$, $g = 1 \ldots N_g$ need to be estimated, where N_g is the number of distinct transformations.

The transformation is typically estimated using the maximum likelihood criterion [33, 69], in which case it is known as maximum likelihood linear regression (MLLR). Recently the maximum *a-posterior* (MAP) criterion has been used to estimate the transformation parameters (MAPLR) [22], allowing the use of prior information in the estimation of the transformation. The minimum classification error criterion (MCE) has also been used [96, 95].

1.1.3 Hybrid adaptation algorithms

Combinations of Bayesian and transformation-based adaptation methods have been shown to combine some of the advantages of the two approaches. Hybrid algorithms using MLLR adaptation followed by MAP adaptation (MLLR-MAP) have been used with much success for speaker adaptation [32, 110] and cross-language adaptation [84]. The MAP adaptation algorithm, followed by the MCE discriminative training procedure (MAP-MCE) [74] has also been used to improve on the MAP and MCE procedures for speaker adaptation.

1.2 Training

Here, only the task-specific training data is available and one can therefore not use adaptation techniques to improve the performance of the resultant system. An example

here would be training a speech recognizer for a new language, where only a small amount of data has been recorded and data from other languages is not available or other complications do not allow the use of adaptation.

1.2.1 Discriminative training

Conventional maximum likelihood (ML) estimation attempts to maximize the likelihood of the training data given the model parameters of the corresponding class. The models from other classes do not participate in the parameter estimation. By maximizing the likelihood of the correct model, but not minimizing the likelihood of other competing models, it cannot be guaranteed that the ML models will optimally discriminate against incorrect classes in recognition and therefore minimize recognition error. Several methods have been proposed to improve this by including discriminative information in the training criterion.

The maximum mutual information (MMI) criterion, which minimizes the class conditional entropy has been used to create a training procedure which is more discriminative [88, 87, 62, 113]. However, as is the case with the ML criterion, MMI does not necessarily minimize the classification error.

The aim of minimum classification error (MCE) training is to correctly discriminate the observations of an HMM for best recognition results and not to fit the distributions to the data. Discriminative training of hidden Markov models (HMMs) using MCE training has been used in several speech recognition tasks with much success. Tasks where MCE training has been used to improve recognition performance include: connected digit recognition [59, 23, 65, 108], the English "E"-set {b,c,d,e,g,p,t,v,z} [23], speaker adaptation [74] and continuous speech [65].

MCE is somewhat prone to overspecialization, especially when training data is limited. Overspecialization is characterized by good recognition performance for the data used

during training, but poor recognition performance for independent testing data. MCE also tends to further emphasize any mismatch between the training and testing sets, resulting in a degradation in testing set performance after a maximum has been reached. Methods of reducing overspecialization in the minimum classification error algorithm will be investigated in Chapter 3.

1.2.2 Bayesian learning

The fundamental concept of Bayesian learning or analysis is that the plausibilities of alternative hypotheses are represented by probabilities, with inference being performed by evaluating these probabilities. The result of Bayesian learning is a probability distribution over model parameters that expresses our beliefs regarding how likely the different model parameter values are.

Given a vector $\mathbf{y} = (y_1, \ldots, y_n)$ of n observations, we have the conditional probability distribution $P(\mathbf{y}|\theta)$, which depends on the k parameters $\theta^T = (\theta_1, \ldots, \theta_k)$. To start the process of Bayesian learning we define a prior distribution $P(\theta)$ for the parameters θ. Using Bayes' rule, the conditional distribution of θ given the observed data (posterior distribution) is

$$P(\theta|\mathbf{y}) = \frac{P(\mathbf{y}|\theta)P(\theta)}{P(\mathbf{y})}. \tag{1.6}$$

The prior distribution is an important part of any Bayesian method, as it expresses our knowledge about the distributions prior to any data being observed. Using the prior distribution $P(\theta)$ and likelihood $P(\mathbf{y}|\theta)$ we can calculate the posterior distribution $P(\theta|\mathbf{y})$ which is then used to classify an unknown observation. Note that $P(\mathbf{y})$ is a normalization term and is usually ignored.

To classify an unknown observation, we integrate the predictions of the model with

respect to the posterior distribution. This is typically a non-trivial task and the integral must either be numerically computed or simplified by approximating the posterior using some parametric form. An approximation which is often used assumes that the posterior is well approximated by a Normal distribution [72, 54, 58]. Assuming such a simple parametric form allows the integral to be easily computed. Such an approximation has the disadvantage that a complex multi-modal posterior distribution cannot be accurately approximated.

Markov chain Monte Carlo methods [36, 37, 38, 39, 26, 79, 111] can be used to numerically integrate the above model prediction with respect to the posterior distribution, and have been used for this purpose in the field of neural networks by Neal [82]. These methods make no assumption concerning the form of the posterior distribution.

The maximum *a-posteriori* (MAP) estimation method has been used extensively in speech recognition and can be considered an approximate Bayesian learning procedure if the posterior distribution is sufficiently peaked about its mode (assuming a single mode).

Bayesian learning allows us to use more complex models when little training data is available (as compared to point estimate techniques). The usage of Bayesian learning using a Markov chain Monte Carlo algorithm will be investigated in Chapter 5.

1.3 Problem statement

Conventional estimation algorithms (such as maximum likelihood estimation) rely on a reasonable amount of training data to give accurate parameter estimates. The accuracy of the parameter estimate is directly related to the recognition performance of the speech recognition system and is therefore of extreme importance. Overtraining, the phenomenon where training set performance is better than the performance for an independent testing set, is also more prevalent when training data is limited (it is

easier for the model to become too specialized).

As mentioned, the collection of large speech databases is an expensive and time consuming task. As a result, it is not always possible to collect, segment and annotate large databases for every task, language and dialect. The sparse training data problem is therefore a real and important problem that must be addressed. This thesis therefore investigates and proposes several techniques which improve the recognition accuracy for sparse training data scenarios.

1.4 Organization of this thesis

This thesis is organized as follows.

In Chapter 2, the relevant hidden Markov model theory is reviewed. The hidden Markov model speech recognition system is also briefly described. The speech corpora used to experimentally evaluate the work in this thesis are described and relevant results from the literature are reported. Finally, overtraining is discussed in terms of the bias/variance problem.

Chapter 3 introduces minimum classification error (MCE) training. Overtraining is discussed within the MCE framework, and several modifications which limit overtraining are proposed. Various aspects of the MCE algorithm and the proposed modifications are discussed and experimentally evaluated.

Chapter 4 focuses on the application of Bayesian theory to adaptation in continuous speech recognition. The classical maximum *a-posteriori* (MAP) adaptation algorithm of Gauvain and Lee [45] is reviewed. An alternative gradient-based method of obtaining the MAP estimate, which does not use a parametric prior distribution, is introduced. A Bayesian inspired modification to the MCE training procedure is proposed. This method effectively tries to obtain the MAP point of the correct classification probability

distribution of the parameters. Finally, the three different methods discussed in this chapter are experimentally compared.

In Chapter 5, Bayesian learning is introduced and an implementation for continuous speech recognition is discussed. Monte Carlo methods relevant to this work are introduced and discussed. The implementation of Bayesian learning within an HMM framework is described. The resultant system is tested for three situations where limited data is available for training purposes.

Finally, in Chapter 6, the discussions and results of previous chapters are summarized and conclusions are made. Suggestions for future research are also given.

1.5 Contributions of this thesis

The contributions of this thesis are,

1. New modifications to the MCE algorithm are proposed in Chapter 3. These modifications limit the effect of overtraining which is prevalent when using MCE training.

2. A new gradient-based MAP adaptation algorithm (GMAP) that does not make any assumptions concerning the form of the prior distribution is proposed in Chapter 4. This algorithm is shown to outperform the standard MAP approach of Gauvain and Lee [45] for the conditions tested.

3. A new MCE based MAP adaptation algorithm is proposed and tested in Chapter 4. This algorithm too is shown to work better than the standard MAP approach, as well as being better than standard MCE.

4. Bayesian learning is introduced. An original implementation of Bayesian learning for hidden Markov model speech recognition is introduced and discussed. This

is, to the author's knowledge, the first time that Bayesian learning using Markov chain Monte Carlo has been used for hidden Markov model speech recognition.

Chapter 2

Background

In this chapter the basic hidden Markov model (HMM) theory and notation is presented. The implementation details of the HMM software, as well as the speech datasets used in this work are also described. Finally, the bias/variance problem is discussed, relating model complexity to overtraining.

2.1 Hidden Markov models

This section documents the relevant HMM theory, as well as any implementation specific details. The configuration of the base system is also described.

The Hidden Markov model Toolkit for Speech Recognition (HMTSR) used in this thesis was developed by the author and Nieuwoudt [83] during their Ph.D. studies. The toolkit is used by several post-graduate students in the Speech Recognition group at the University of Pretoria.

It is not realistic to present a thorough review of hidden Markov modeling theory in this chapter; the reader is therefore referred to books such as that of Rabiner and Juang

[90] or their article "An introduction to Hidden Markov Models" [94, 92].

2.1.1 Feature extraction

Thirteen Mel-frequency cepstral coefficients (MFCCs), along with their first and second order differentials are used. The following describes the feature extraction process:

- A first order filter is used to pre-emphasize the speech signal [90, p. 112], to spectrally flatten the signal and to limit finite precision effects later in feature extraction. The filter transfer function used is

$$H(z) = 1 - a \cdot z^{-1}, \tag{2.1}$$

  where $a = 0.98$ was empirically found to work well.

- The preemphasized speech signal is blocked into frames [90, p. 113] of length 16ms, with overlap of 6ms. The frames are therefore 10ms apart. The number of samples (N_s) is determined by the block length (time) and the sampling rate. A sampling rate of 16kHz would therefore result in a block of 256 samples.

- A Hamming window [90, p. 114] is used to minimize signal discontinuities at the beginning and end of each frame. A Hamming window has the following form:

$$w(n) = 0.54 - 0.46cos(\frac{2\pi n}{N_s - 1}) \qquad 0 \leq n \leq N_s - 1. \tag{2.2}$$

- The power spectrum of the windowed frame is calculated using the fast Fourier transform.

- The power spectrum is filtered using N_f mel-spaced filters [90, p. 183-190]. The filters have triangular bandpass frequency responses. The number of filters is set using the following formula $N_f = round(0.0015 \cdot S_r)$, where S_r is the sampling

rate. A sampling rate of 16kHz therefore results in 24 mel-spaced filters being used.

- The discrete cosine transform (DCT) of the natural logarithm of the 24 filter outputs is computed.
- The first 13 DCT coefficients are kept as the MFCCs.
- A second-order polynomial fitting [90, p. 194] of 5 consecutive MFCCs is used to estimate the first and second order derivatives of the MFCCs. This incorporates temporal information (external to the frame) into the features.

Note that the values in the above feature extraction process were determined empirically and will not necessarily perform best in all circumstances.

2.1.2 Continuous density hidden Markov models

A continuous density HMM is characterized by the following:

1. The number of states N. The individual states are labeled as $\{1, 2, \ldots, N\}$, and the state at time t is denoted as q_t.
2. The number of mixture components per state M. Figure 2.1 shows a continuous density HMM with 3 states ($N = 3$) and 4 Gaussian mixture components ($M = 4$).
3. The state-transition probability distribution $A = \{a_{ij}\}$, where a_{ij} is the probability of being in state j at time $t + 1$ after having been in state i at time t, i.e.

$$a_{ij} = P(q_{t+1} = j | q_t = i), \qquad 1 \leq i \leq N, 1 \leq j \leq N + 1. \tag{2.3}$$

Note that the probability of *leaving* an HMM from state i is given by $a_{i(N+1)}$.

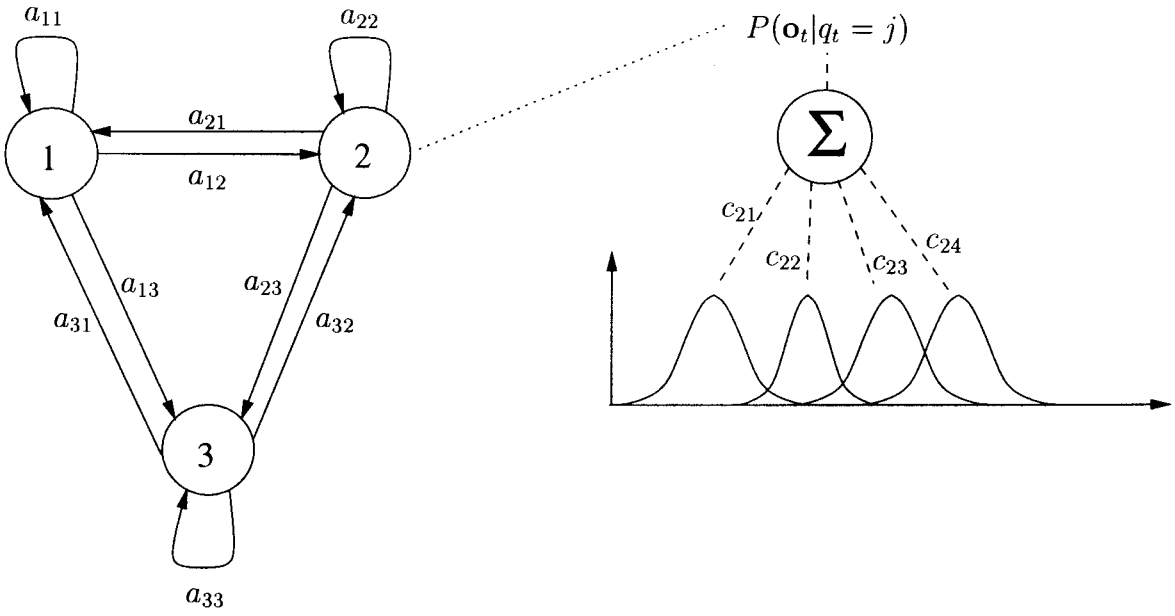


Figure 2.1: An $N = 3$ state $M = 4$ Gaussian mixture HMM.

4. The observation probability distribution $B = \{b_j(\mathbf{o}_t)\}$ for the observation $\mathbf{o}_t$, where

$$b_j(\mathbf{o}_t) = P(\mathbf{o}_t|q_t = j) \qquad 1 \leq j \leq N. \tag{2.4}$$

For a continuous density HMM the observation probability is represented as a finite mixture of the form

$$b_j(\mathbf{o}) = \sum_{k=1}^{M} c_{jk}\mathcal{N}(\mathbf{o}, \mu_{jk}, \mathbf{\Sigma}_{jk}). \tag{2.5}$$

5. Initial state probability distribution $\pi = \pi_i$, where

$$\pi_i = P(q_1 = i), \qquad 1 \leq i \leq N. \tag{2.6}$$

A complete specification of an HMM includes the parameters N, M, a_{ij}, π_i, and the mixture parameters c_{jk}, μ_{jk} and Σ_{jk}. The complete parameter set above of an HMM will be represented as

$$\theta = (A, B, \pi). \tag{2.7}$$

The MFCC features are largely uncorrelated. A diagonal covariance matrix (Σ_{jk}) is therefore used in this work, which greatly reduces the number of parameters that are

used.

Left-to-right hidden Markov models limit transitions to forward transitions only, i.e. $a_{ij} = 0 \quad \forall\, j < i$. Left to right hidden Markov models with no skipping transitions are used in this work where $a_{ij} = 0 \quad \forall\, j \neq i$ and $j \neq (i+1)$.

The probability of an observation sequence $\mathbf{O} = (\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_T)$ given the model θ ($P(\mathbf{O}|\theta)$) can be obtained by summing over all possible state sequences $\mathbf{q} = (q_1 q_2 \ldots q_T)$,

$$P(\mathbf{O}|\theta) = \sum_{\text{all } \mathbf{q}} P(\mathbf{O}, \mathbf{q}|\theta) = \sum_{\text{all } \mathbf{q}} P(\mathbf{O}|\mathbf{q}, \theta) P(\mathbf{q}|\theta). \tag{2.8}$$

Assuming statistical independence of observations, we can write $P(\mathbf{O}, \mathbf{q}|\theta)$ as

$$\begin{aligned} P(\mathbf{O}, \mathbf{q}|\theta) &= \pi_{\mathbf{q}_0} b_{\mathbf{q}_1}(\mathbf{o}_1) \prod_{t=2}^{T} \left(a_{\mathbf{q}_{t-1}\mathbf{q}_t} \cdot b_{\mathbf{q}_t}(\mathbf{o}_t) \right) \\ &= \prod_{t=1}^{T} \left(a_{\mathbf{q}_{t-1}\mathbf{q}_t} \cdot b_{\mathbf{q}_t}(\mathbf{o}_t) \right), \end{aligned} \tag{2.9}$$

where $a_{\mathbf{q}_0\mathbf{q}_1}$ is taken to be $\pi_{\mathbf{q}_0}$.

The Viterbi algorithm is used to find the single best state sequence $\mathbf{q} = (q_1 q_2 \ldots q_T)$ and its probability $P(\mathbf{O}, \mathbf{q}|\theta)$, for a given observation sequence $\mathbf{O} = (\mathbf{o}_1 \mathbf{o}_2 \ldots \mathbf{o}_T)$. The Forward-Backward procedure is used to determine the probability of an observation sequence given the model θ, i.e., $P(\mathbf{O}|\theta)$. Due to efficiency considerations, the best state sequence is used for recognition and some training procedures, as opposed to using all possible state sequences.

2.1.3 Training

The hidden Markov models are trained in at least three phases, namely

1. initialization,
2. segmental training
3. maximum likelihood training.

Embedded training is often used after the above training steps. A short discussion of the training procedures and related algorithms follows:

Initialization The HMM parameters are initialized as follows:

- The transition probabilities, initial probabilities, number of states and Gaussians per state are set by the user in a configuration file. In this work, the transition probabilities were always initialized as

$$a_{ij} = \begin{cases} 0.9 & j = i, \\ 0.1 & j = i + 1, \\ 0 & \text{otherwise}, \end{cases} \tag{2.10}$$

  with the initial probabilities being initialized as

$$\pi_i = \begin{cases} 1 & i = 1, \\ 0 & \text{otherwise}. \end{cases} \tag{2.11}$$

- The Gaussian mixture weights are initialized as $1/M$, for an M Gaussian mixture state.
- Labeled data is divided into equal sized segments, one segment for each state in the HMM. The speech segments are used to initialize the HMM mixture means and variances by using the segmental K-means algorithm to cluster the features into M clusters (M Gaussian mixture state), the means and variances of which are used to initialize the mixtures for the associated state.

Alternately, in the absence of labeled data, utterances are divided into equal sized blocks using the transcription given. Further training then proceeds using search-based training.

Maximum likelihood training The Baum-Welch method, also known as expectation maximization (EM) [28], is used to iteratively maximize the likelihood $P(\mathbf{O}|\theta)$.

To describe the procedure of segmental re-estimation of the parameters, we first define $\xi(i,j)$, the probability of being in state i at time t and state j at time $t+1$. We also define $\gamma_t(i)$ as the probability of being in state i at time t and $\gamma_t(j,k)$ as the probability of being in state j at time t with the kth mixture component accounting for $\mathbf{o}_t$.

The forward-backward algorithm [90, p. 334] is used to estimate the above probabilities given the current model. The parameter re-estimation formulas used are,

$$a_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{2.12}$$

$$c_{jk} = \frac{\sum_{t=1}^{T} \gamma_t(j,k)}{\sum_{t=1}^{T} \sum_{m=1}^{M} \gamma_t(j,m)} \tag{2.13}$$

$$\mu_{jkl} = \frac{\sum_{t=1}^{T} \gamma_t(j,k) \cdot \mathbf{o}_t}{\sum_{t=1}^{T} \gamma_t(j,k)} \tag{2.14}$$

$$\sigma_{jkl} = \frac{\sum_{t=1}^{T} \gamma_t(j,k) \cdot (\mathbf{o}_t - \mu_{jkl})(\mathbf{o}_t - \mu_{jkl})'}{\sum_{t=1}^{T} \gamma_t(j,k)}. \tag{2.15}$$

Segmental training The forward-backward algorithm used in maximum likelihood training is relatively computationally expensive and slow. A segmental training step [61] is therefore used to quickly obtain a relatively good estimate of the model parameters (prior to using the EM/ML algorithm). The Viterbi algorithm is used to perform a forced-alignment, in which the best state sequence is obtained. The state alignment is then used to reestimate the model parameters. This is often called segmental training, as the observation sequence is segmented (state aligned), with the segments being

used to reestimate the parameters of the associated states. It would, however, only be fair to mention that the forward-backward algorithm does have a stronger theoretical background.

The same re-estimation formulas as that used in the ML/EM algorithm (Eqs. (2.12) to (2.15)) are also used here. The probabilities $\xi_t(i,j)$, $\gamma_t(i)$ and $\gamma(j,k)$ are, however, estimated as follows

$$\xi_t(i,j) = \delta(\mathbf{q}_t - i)\delta(\mathbf{q}_{t+1} - j) \tag{2.16}$$

$$\gamma_t(i) = \delta(\mathbf{q}_t - i) \tag{2.17}$$

$$\gamma_t(j,k) = \delta(\mathbf{q}_t - j)\frac{c_{jk}\mathcal{N}(\mathbf{o}_t, \mu_{jk}, \Sigma_{jk})}{b_j(\mathbf{o}_t)} \tag{2.18}$$

where $\delta()$ is the Kronecker delta function. The Kronecker delta function is defined as

$$\delta(i) = \begin{cases} 1 & \text{if } i = 0 \\ 0 & \text{otherwise.} \end{cases} \tag{2.19}$$

Embedded training It is often advantageous to use a search algorithm (see Section 2.1.5), not only to perform a state alignment, but also HMM alignment. Here a search algorithm is used along with the transcription of the speech units to automatically align (segment) the speech data (HMM alignment). We are therefore not relying on any manually created labels, but allowing the current model to determine where a label should begin and end. The search is limited to the transcription of the relevant utterance. Some optional HMM models are permitted inside the transcription, this is to allow models such as silence to be inserted (by the search), where they might not occur in the transcription. The re-estimation is performed in exactly the same way as done in the segmental training step. In some situations, where some or all of the data is not labeled, this step is essential. For a dataset which is not labeled at all, such as

the TIDIGIT dataset, only the initialization step and embedded training is performed (the other training phases rely on labelled data).

Mixture splitting An alternative that has been found to work well is Gaussian splitting [100, 87]. Here, we initialize only one mixture component per state and then split the mixture component with the highest mixture component weight into two separate Gaussians. This is done at the end of each training iteration, until all allowed mixtures components have a non-zero mixture weight. The Gaussians are split in the direction of maximum variance. This is done by setting the mean and variance of the new Gaussian equal to the existing Gaussian and then moving the mean vectors a small distance away from each other in the direction of maximum variance. The weights of the two new Gaussians are set to be equal to one half that of the original Gaussian.

Figure 2.2 shows a two-dimensional example of a two-component Gaussian mixture where the component with the largest weight (0.7) is split into two new Gaussians, creating a mixture containing three Gaussians. Note that the mean and variance of the new Gaussians will tend to change considerably after the next training iteration. If, during training, a mixture weight were to become zero, the Gaussian associated with the largest weight will be split. This technique has been used for all experiments in this work.

2.1.4 Duration modeling

Hidden Markov models implicitly model state duration probability with a Geometric distribution, i.e.

$$p_i(d) = a_{ii}^{d-1}(1 - a_{ii}), \tag{2.20}$$

which is the probability of d consecutive observations in state i. This exponential state

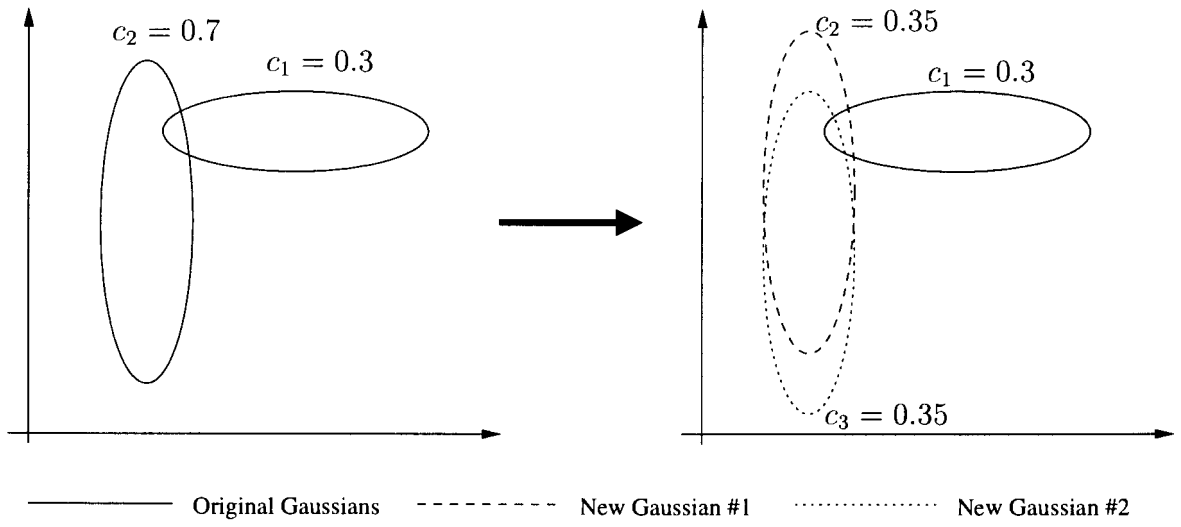


Figure 2.2: Two-dimensional example of mixture component splitting.

duration density is inappropriate for most systems. We would prefer to explicitly model the duration probability in some analytic form. Initial duration modeling approaches [90] assigned each state a discrete duration probability, which was incorporated into the search algorithm. These algorithms were, however, computationally expensive. Post-processing approaches [91, 93], which were more computationally efficient, added the duration contribution to the Viterbi metric after candidate paths had been identified. If the best path is not one of the candidate paths, then this method fails.

Burshtein [16] proposed a duration modeling approach, which adds the duration metric at each frame transition in the Viterbi algorithm.

The gamma distribution,

$$p(d) = \frac{\alpha^{\beta}}{\Gamma(\beta)} x^{\beta-1} e^{-\alpha x}, \quad 0 < d < \infty, \tag{2.21}$$

is often used to model the duration density, with $\alpha > 1$ and $\beta > 0$.

The modified Viterbi algorithm keeps track of the duration $d_s(t)$ of each state s at time t. Letting M_i denote the duration, at which the gamma distribution of state i is

a maximum, then the duration penalty P_{ij} of making a transition from state i at time t to state j at time t+1 is

$$P_{ij} = \begin{cases} 0 & d_i(t) < M_i \text{ and } i = j \\ log(\frac{d_i(t+1)}{d_i(t)}) & d_i(t) \geq M_i \text{ and } i = j \\ log(d_i(t)) & d_i(t) < M_i \text{ and } i \neq j \\ log(M_i) & d_i(t) \geq M_i \text{ and } i \neq j. \end{cases} \tag{2.22}$$

Auto-transitions are therefore not penalized before the duration is M_i. After the duration M_i, we penalize gradually. Upon exit from the state, the overall duration metric is $log(d_i(t))$, which is as it should be. Word duration modeling is implemented in much the same way.

The duration model parameters are estimated using the Viterbi state-alignment. After determining the mean and variance of the duration from the state alignment, the parameters α and β are estimated for each state using Eqs. (A.14) and (A.15). A similar approach to duration modeling was also proposed by Du Preez [35]. Burshtein's approach has been implemented and is used, where indicated, in this thesis.

2.1.5 Search

Continuous speech recognition requires the segmentation and labeling of continuous speech. Fortunately, there are efficient methods of segmenting and labeling continuous speech at word and phoneme levels. Various (search) algorithms exist, each with advantages and disadvantages. However, only those algorithms implemented and used in this work will be described here.

Grammar networks and language models A grammar network/language model is used to determine which transitions may be taken, or what probability there is of taking a certain transition. A grammar is an explicit set of rules limiting which models/words/phonemes may follow others. The grammar is implemented as a finite state network (FSN), which is created from a text based grammar. Note that each FSN node is associated with a single HMM, but there may be multiple nodes associated with the same HMM.

Trellis search The trellis search algorithm [106] is a frame-synchronous implementation of the Viterbi search algorithm, which allows transitions between models or grammar nodes.

N-best search It is often necessary to obtain the N-best HMM sequences. The N-best strings are often used to obtain a confidence measure for a recognition output. MCE training uses the N-best strings to obtain a measure of misclassification. If the problem is simple enough, the N-best HMM sequences can be obtained by aligning all possible HMM sequences. This is, however, not feasible for most, if not all, speech recognition applications. As a result, efficient N-best search algorithms have been developed [106, 103]. The disadvantage, is that none of these algorithms are guaranteed to give the true N-best sequences (although most times there will only be small differences in probability and not the sequences).

The N-best search implemented and used in this work is the tree-trellis algorithm proposed by Soong and Huang [106]. The search comprises two stages, namely

1. Modified Viterbi algorithm (Trellis search algorithm)

   The standard trellis search algorithm is modified, to store a partial path map. The partial path map contains the likelihood scores of all partial paths leading to every grammar node from all previous nodes.

2. A^* Tree search

   After the trellis search algorithm has been performed, a backward tree search is started from the terminal node. This part of the search is done time asynchronously. The search tree is implemented using the A^* algorithm [99, 86]. The A^* search restricts the search by using an admissible heuristic h, such that h never overestimates the cost to reach the goal. Here, we use the partial path map, which gives the exact cost.

2.2 Overtraining

Overspecialization (or overtraining) occurs in most training algorithms where a finite number of examples are available for training. If the training data set was perfectly representative of the test set (this would only truly occur when an infinite number of training examples were available), there would be no difference between training set and testing set performance. However, in practice data sets are limited in size and the test set performance tends to be worse than the training set performance. This is a result of the model becoming too specialized and not generalizing well.

Model complexity influences overtraining, especially when little training data is available. This phenomenon can be described in terms of the bias/variance problem.

2.2.1 The Bias/Variance Dilemma

Here, due to its less complex and easily interpreted nature, the bias/variance problem will be discussed within a regression framework. It is, however, just as relevant to classification problems.

The regression problem is to create a function $f(\mathbf{x}|\lambda)$ using a training set $\mathcal{D}$, with $\mathcal{D} = (x_1, y_1), \ldots, (x_n, y_n)$, where we wish to estimate $\mathbf{y}$ for an observation $\mathbf{x}$. Typically,

the function f is fixed and we wish to merely estimate the function parameters λ. Given the data ($\mathcal{D}$), and $\mathbf{x}$, an appropriate measure of the suitability of $f(\cdot|\lambda)$ as a predictor of $\mathbf{y}$ is the mean-squared error (MSE), or

$$E[(\mathbf{y} - f(\mathbf{x}; \mathcal{D}))^2|\mathbf{x}, \mathcal{D}].$$

It can easily be shown [46] that

$$\begin{aligned} E[(\mathbf{y} - f(\mathbf{x}; \mathcal{D}))^2|\mathbf{x}, \mathcal{D}] &= E[(\mathbf{y} - E[\mathbf{y}|\mathbf{x}])^2|\mathbf{x}, \mathcal{D}] \\ &\quad + (f(\mathbf{x}; \mathcal{D}) - E[\mathbf{y}|\mathbf{x}])^2. \end{aligned} \tag{2.23}$$

The expectation $E[(\mathbf{y} - E[\mathbf{y}|\mathbf{x}])^2|\mathbf{x}, \mathcal{D}]$ is not dependent on the data or on the estimator f. The distance $(f(\mathbf{x}; \mathcal{D}) - E[\mathbf{y}|\mathbf{x}])^2$ therefore measures the effectiveness of f as a predictor of $\mathbf{y}$. The mean-squared error of f as an estimator of the regression $E[\mathbf{y}|\mathbf{x}]$ is

$$E_{\mathcal{D}}[(f(\mathbf{x}; \mathcal{D}) - E[\mathbf{y}|\mathbf{x}])^2], \tag{2.24}$$

where $E_{\mathcal{D}}$ is the expectation with respect to the training set $\mathcal{D}$, i.e. the average over the ensemble of possible datasets $\mathcal{D}$. Equation (2.24) can be rewritten in terms of bias and variance [46],

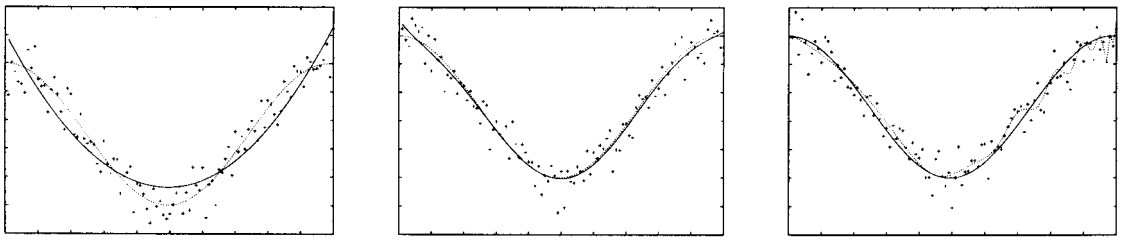
$$\begin{aligned} E_{\mathcal{D}}[(f(\mathbf{x}; \mathcal{D}) - E[\mathbf{y}|\mathbf{x}])^2] &= (E_{\mathcal{D}}[f(\mathbf{x}; \mathcal{D})] - E[\mathbf{y}|\mathbf{x}])^2 \\ &\quad + E_{\mathcal{D}}[(f(\mathbf{x}; \mathcal{D}) - E_{\mathcal{D}}[f(\mathbf{x}; \mathcal{D})])^2]. \end{aligned} \tag{2.25}$$

The term $(E_{\mathcal{D}}[f(\mathbf{x}; \mathcal{D})] - E[\mathbf{y}|\mathbf{x}])^2$ is the bias of the estimator and measures any systematic tendency for it to give the incorrect answer. An estimator (or classifier) is said

to be biased when the bias term is non-zero. The term $E_{\mathcal{D}}[(f(\mathbf{x};\mathcal{D}) - E_{\mathcal{D}}[f(\mathbf{x};\mathcal{D})])^2]$ is the variance in the estimator error, and measures the sensitivity of the estimator to any randomness in the training examples.

The bias and variance of an estimator are typically affected by, among others, model type, model complexity and the parameter estimation algorithm. Unfortunately, reducing the bias typically increases the variance (and *vice versa*). Reducing the sum of the bias and variance (or mean-squared error of f), therefore generally requires a trade-off between their contributions.

The trade-off between bias and variance is usually optimized by varying the complexity of the model. This trade-off between bias and variance can be illustrated using a one-dimensional regression problem. Figure 2.3 shows one such example. In this case, a polynomial of degree n is fitted to the noisy data by minimizing the mean-squared error. Results using $n = 2$, $n = 5$ and $n = 50$ are shown.



$MSE = 0.0107$ $MSE = 0.0005$ $MSE = 0.0021$

(a) (b) (c)

Figure 2.3: One hundred observations of the raised cosine function $(0.5cos(10x/pi) + 1)$ plus noise. Noise has zero-mean Gaussian distribution, with standard deviation 0.1. The solid curve is the target function, and dotted curve the polynomial fit of degree n, with (a) $n = 2$, (b) $n = 5$ and (c) $n = 50$.

The polynomial fit in (a) is relatively poor, and is a result of the inability of the model (or function) to represent the underlying process. The solution in (a) has a large mean-squared error due mainly to the bias term. The polynomial fitted in (c) has a high degree n, and has started fitting the noise; here the variance term accounts for most of the mean-squared error. The polynomial fit shown in (b) is the result of a model which

is complex enough to fit the true underlying process (low bias), yet simple enough such that the noise in the data is not modeled as well (low variance).

The optimal choice of model complexity will also vary with the amount of available training data. More training data will reduce the variance and so more complex models can be used. However, when there is relatively little data available, less complex models will be preferred.

The principles in the above discussion are incorporated in *Occam's Razor*. Simply stated, *Occam's Razor* is a principle that states that unnecessarily complex models should not be preferred to simpler ones. The bias/variance dilemma is often problematic when using single point estimates in sparse training data situations. The effects of the bias variance problem will be encountered in Chapter 4 where, for certain sparse data scenarios, less complex models are preferable. In Chapter 5 the usage of Bayesian learning is investigated. Bayesian learning reduces the variance of the estimate or solution and therefore results in more complex models being preferred over less complex ones in such sparse data scenarios.

2.3 Experimental procedure

The main goal of the experiments designed in this study is to investigate the relative effectiveness of the algorithms proposed in comparison with conventional algorithms. This work therefore attempts to keep as much in common for all other aspects of the speech models associated with both the conventional and the proposed techniques and to keep the recognizer structure as simple as possible. The basic HMM system described in Section 2.1 has been used throughout.

There is little point in presenting discrete phoneme recognition (phoneme classification) results as this can be misleading. A phoneme classifier that only chooses the most frequently observed phonemes will tend to perform well for phoneme classifica-

tion. Such a classifier will, however, not necessarily work well for other tasks such as continuous word or phoneme recognition. Continuous phoneme recognition results are therefore reported throughout this thesis. The recognition accuracy of the system is reported, where accuracy is defined as:

$$Accuracy = \frac{Phones - Subs - Dels - Ins}{Phones}, \tag{2.26}$$

where $Phones$ refers to the number of phones in the correct transcription, $Subs$ the number of substitutions, $Dels$ the number of deletions and Ins the number of insertions. Error rates reported are simply $100\% - Accuracy$.

It is necessary to determine whether the recognition accuracy of a new system is better than that of an existing baseline or reference system. A test of significance can therefore be performed to determine whether this is probably true or not. We must therefore decide between the two hypotheses:

H_0: $P_n > P_b$,

H_1: $P_n <= P_b$,

where P_n is the recognition accuracy of the new system and P_b is the baseline or reference accuracy. A one-tailed test is used, since we are interested in determining whether the improvement in recognition accuracy is better than a reference system performance.

The maximum probability with which we would be willing to risk the error of rejecting a hypothesis when we should have accepted it is called the level of significance. If, for example, a level of significance of 0.01 were attained, then we would be 99% confident that we had made the correct decision in accepting the hypothesis.

Table 2.1 details the absolute improvements in recognition accuracy required to attain

a 0.05 and 0.01 level of significance for the three speech databases used (described in Section 2.4). The baseline accuracy is assumed to be 50%, i.e. the worst case improvement required (a 70% baseline performance, for example, will require a smaller improvement to reach the same level of significance).

Table 2.1: Improvements in recognition accuracy required to attain a 0.05 and 0.01 level of significance for the three speech databases used. The baseline accuracy is assumed to be 50%. The number of phonemes in the relevant testing sets, used to determine the significance level, are also listed.

Database	Phonemes	Level of significance	
		0.05	0.01
TIMIT	59858	0.34%	0.48%
TIDIGITS	28330	0.49%	0.69%
SUNSpeech	9026	0.87%	1.23%

The improvements in accuracy required (Table 2.1) are typically smaller than one percent, but greater than 0.1%. Results will therefore be specified to one decimal place. In this work, improvements that are significant to a level of 0.01 will simply be referred to as *significant*.

2.4 Speech datasets

This section describes the speech corpora used in this thesis. Three different datasets are used, namely: TIMIT, TIDIGITS and SUNSpeech.

2.4.1 TIMIT

The TIMIT [4] corpus of read speech was designed to provide speech data for the acquisition of acoustic-phonetic knowledge and for the development and evaluation of automatic speech recognition systems. TIMIT contains a total of 6300 sentences, 10

sentences spoken by each of 630 speakers from 8 major dialect regions of the United States.

The sentences found in TIMIT consist of 2 dialect "shibboleth" sentences, 450 phonetically compact sentences, and 1890 phonetically diverse sentences. The dialect sentences (SA sentences) were meant to expose the dialectal variants of the speakers and were read by all 630 speakers. The phonetically-compact sentences were designed to provide a good coverage of pairs of phones, while the phonetically-diverse sentences (the SI sentences) were selected from existing text sources so as to add diversity in sentence types and phonetic contexts. Table 2.2 summarizes the sentences found in the TIMIT database.

Table 2.2: Summary of sentences found in TIMIT

Sentence type	Unique sentences	Number of speakers	Total	Sentences per speaker
Dialect (SA)	2	630	1260	2
Compact (SX)	450	7	3150	5
Diverse (SI)	1890	1	1890	3
Total	2342		6300	10

Following convention the standard TIMIT 39-phone set is used. The 63 phones found in the TIMIT database are reduced to 39 by combining models as done by Lee [68].

Training and testing sets

The TIMIT dataset is used in experiments in Chapters 3, 4 and 5. In Chapter 3 overtraining is investigated using the suggested training and testing sets. Chapters 4 and 5 however, present gender adaptation experiments and limited training data experiments using TIMIT.

A reduced female training set is used for this purpose, so as to create a scenario where

there is limited training data for female speakers. The small female training set consists of speech data from two speakers from each of the eight dialect regions. No sentence text appears in both the training and test sets. A small gender independent training set will be required for testing the algorithms presented in Chapters 3 and 5. One such set has been created by randomly selecting two male and two female speakers from each of the eight dialect regions found in the TIMIT dataset.

Table 2.3 describes the different training and test sets used. Gender specific testing sets are used in Chapters 4 and 5.

Table 2.3: Description of TIMIT training and testing sets used

Description	Label	Number of speakers			Number of Sentences	Duration (minutes)
		Male	Female	Total		
Training sets:						
Full (standard)	T	326	136	462	4620	236.5
Male	T_M	326	0	326	3260	165.2
Female	T_F	0	136	136	1360	71.3
Female-small	T_{FS}	0	16	16	160	8.2
Small (gender indep.)	T_S	16	16	32	320	16.1
Testing set (standard)		112	56	168	1680	86.4

Use in literature

The TIMIT dataset has been used extensively in the speech recognition literature to experimentally test algorithms and hypotheses. The following summarizes a few such cases, so as to ensure that the reader has a fair idea of what performance can be expected for TIMIT. Only system configurations similar to that used here are reported.

Rathinavela and Deng [96] investigated the usage of state-dependent linear transforms of Mel-warped DFT features. The authors performed discrete (not continuous)

phoneme recognition to test ML and MCE trained HMMs, as well as their "optimum-transformed HMM". Table 2.4 summarizes the results reported for simple left-to-right 3 state, 5 mixture HMM models.

Table 2.4: Summary of TIMIT results reported in [96]

Model/Training	Phonetic classification rate (%)
ML-HMM	59%
MCE-HMM	66%
MCE-THMM	69%

McDermott [75] used the TIMIT database to evaluate the effectiveness of MCE for continuous speech recognition. Table 2.5 summarizes the TIMIT results presented by McDermott in [75].

Table 2.5: Summary of TIMIT accuracy results reported by McDermott [75]

Number of mixtures	ML	MCE	ML+bigrams	MCE+bigrams
1	48.8		56.0	61.0
4	55.0	61.9	62.4	66.2
8	57.1	62.7	64.8	67.3
16	59.9	63.2	66.8	68.7

Yuk and Flanagan [119] investigate the use of neural network based adaptation methods applied to telephone speech recognition using TIMIT and NTIMIT [56]. Recognition accuracy of 62.2% is reported for TIMIT using their base system, a 3 state left-to-right mono-phone HMM with 30 Gaussian distributions per state.

Moreno and Stern [80] compared speech recognition accuracy for high quality recorded speech and speech as it appears over long-distance telephone lines. The performance of the CMU SPHINX system was compared for the TIMIT and NTIMIT [56] databases. Recognition accuracy of 52.7% was reported for the TIMIT test set.

2.4.2 TIDIGITS

The TIDIGITS [70] was designed and collected for the purpose of designing and evaluating algorithms for speaker-independent recognition of connected digit sequences. The corpus contains read utterances from 326 speakers (111 men, 114 women, 50 boys, and 51 girls) each speaking 77 digit sequences. The data was collected in a quiet environment and digitized at 20 kHz.

The digit sequences are made up of the digits: "zero", "oh", "one", "two", "three", "four", "five", "six", "seven", "eight", and "nine". The 77 digit sequences spoken by each of the speakers can be broken up as follows: 22 single-digit sequences (2 of each of the 11 digits) and 11 each of randomly generated 2,3,4,5 and 7-digit sequences.

Training and testing sets

The database is divided into two subsets, one to be used for algorithm design and the other for evaluation. The division yielded speaker independent training and testing sets, each containing half of the male and female speakers. The boys and girls test and training utterances are not used in this work.

As with TIMIT, the TIDIGIT dataset is used for gender adaptation and reduced data experiments in later chapters. It is for these experiments that reduced subsets are created for female speakers in the dataset. A reduced speaker set (T_{WS}) is created, using four randomly chosen female speakers, using all 77 digit sequences per speaker. The above set is further reduced by using only 7 digit sequences per speaker (2 single-digits and one each of 2,3,4,5 and 7 digit sequences). Table 2.6 presents the training and testing sets used for the TIDIGITS corpus.

Table 2.6: Training and testing sets used with the TIDIGIT database

Description	Label	Number of speakers			Digit sequences	Duration (minutes)
		Male	Female	Total		
Training sets:						
Full (standard)	T	55	57	112	8624	253.4
Man	T_M	55	0	55	4235	121.9
Woman	T_W	0	57	57	4389	131.5
Woman-small	T_{WS}	0	5	5	385	10.1
Woman-very-small	T_{WVS}	0	5	5	35	55 s
Testing sets:						
Full		55	57	112	8623	254.4
Man		55	0	55	4235	123.1
Woman		0	57	57	4389	131.3

Use in literature

Normandin [87] proposed an approach for splitting Gaussian mixture components based on maximum mutual information estimation (MMIE) training. Experiments using the TIDIGITS dataset were conducted. Recognition word (digit) error rates of between 1.6% (1 mixture per state) and 1.0% (16 mixtures per state) were obtained for models with variable numbers of states. Utilizing their mixture splitting algorithm, however, resulted in a digit error rate of 0.71%.

Jiang *et al.* [57] investigated a new Bayesian predictive classification (BPC) approach for robust speech recognition where a mismatch between training and testing conditions occurred. TIDIGITS was used, along with other datasets, to test their new approximate BPC algorithm. Using a 10 state, 10 mixture per state continuous density HMM, digit error rates of 2.2% and 2.4% were reported for their baseline system and new BPC system respectively, when using the standard TIDIGITS dataset.

2.4.3 SUNSPEECH

The SUN Speech database was compiled by the Department of Electrical and Electronic Engineering of the University of Stellenbosch to contain phonetically labelled speech in both English and Afrikaans. The data was recorded in a controlled environment, with 12 bit resolution and a 16kHz sampling rate. Sixty sentences comprising four sentence sets were chosen to exhibit the diversity of phonemes in the two languages. Details of the number of speakers and the number of sentences spoken by each group of speakers are given in Table 2.7.

Table 2.7: Description of SUN Speech database: number of male and female speakers and total number of speakers for each sentence set

Language	Number of speakers			Sentence set	Number of sentences
	Male	Female	Total		
Afrikaans	24	16	40	1	10
	18	12	30	2	10
English	33	17	50	3	20
	22	4	26	4	20

A total of 59 phonetic categories, including both a *silence* and *unknown* category, were used to segment both the Afrikaans and the English speech. It was attempted to assign the labels phonetically, i.e. according to the sound produced, rather than phonologically assigning the labels, i.e. according to what was supposed to be said.

Training and testing sets

The SUN Speech database is used for cross-language adaptation experiments in this work. Subdivision of the dataset was therefore dictated by the requirements for cross-language adaptation.

A speaker independent (SI), sentence independent division of the Afrikaans data can

be obtained by using data from the first sentence set for training and data from the second sentence set for testing. We however, need to create a smaller training subset, so as to recreate a scenario where extremely little Afrikaans training data is available.

The database is not entirely consistent in that some speakers, but not all, spoke sentences from more than one sentence set. Those speakers who spoke all of the Afrikaans sentences were used to create a reduced Afrikaans set. This has the added utility of creating a speaker-dependent (SD) test set; although this set is not used in this thesis. The reduced Afrikaans training set will be referred to as the "adaptation set", as it is used as such in many of the experiments.

The full English set is used for training purposes (it is not required for testing). Table 2.8 gives the details of the subdivision of the database into training and testing sets.

Table 2.8: Details of SUNSpeech training and testing sets used

Language	Description	Label	Number of Speakers			Sentence set	Duration (minutes)
			Male	Female	Total		
English	Training	E	55	21	76	3 and 4	135.5
Afrikaans	Training set	A	23	16	39	1	22
	Training subset	A_S	2	6	8	1	5.5
	SD test set		2	6	8	2	7.7
	SI test set		14	1	15	2	13

Use in literature

The usage of this database in the speech recognition literature is somewhat limited. Waardenburg *et al.* [115] investigated the isolated recognition of stop consonants using HMMs. More recently, Nieuwoudt and Botha [84, 85] investigated cross-language usage of acoustic information using this dataset, where continuous word recognition results were presented. To the authors knowledge, no continuous phoneme recognition results

have been published for this dataset.

Chapter 3

Minimum classification error training

Although there are several learning algorithms (such as *maximum mutual information* and *minimum discriminative information*) which can be classified as discriminative training techniques, this chapter will exclusively describe Minimum classification error learning and several modifications thereof.

3.1 Introduction

Conventional maximum likelihood (ML) estimation attempts to maximize the likelihood of the training data given the model parameters of the corresponding class. The models from other classes do not participate in the parameter estimation. By maximizing the likelihood of the correct model, but not minimizing the likelihood of other competing models, it cannot be guaranteed that the ML models will optimally discriminate against incorrect classes in recognition. Maximum likelihood estimation can be problematic in situations where the distribution of the data to be recognized

is significantly different from the distribution of the model, as noted in the literature [81, 14, 75].

A theory of error-corrective training for pattern classification was first proposed by Amari [2], where an adaptive procedure was developed and shown to converge to a local minimum of the classification error function.

Franco and Serralheiro [42] proposed a training procedure which aims explicitly at reducing the recognition error and increasing discrimination between classes. The algorithm is based on a criterion function which is the quadratic error between target state probabilities and the *a-posteriori* state probabilities given the training data. The target state probabilities are forced to be one for the correct state and zero for incorrect states. This criterion function was then used to adjust HMM parameters heuristically to improve the training set recognition rate.

Chen and Soong [19] introduced an N-best candidates based frame-level discriminative training algorithm based on an N-best tree-trellis algorithm [106]. A frame-level loss function was defined and minimized using the gradient descent method. The loss function was defined as a half-wave rectified log-likelihood difference between the correct and selected competing hypotheses. The loss function is accumulated over all training utterances. Their algorithm was tested using a connected Chinese digit recognition experiment and a large vocabulary isolated word experiment. Significant improvements over traditional ML were reported, with the string error rate being reduced from 17.0% to 10.8% for the connected digit experiment and the isolated word recognition error rate being reduced from 7.2% to 3.8%.

Maximum mutual information (MMI) A more formal, information theoretic approach based on the maximum mutual information criterion has been used to train HMM based speech recognition systems [88, 87, 62, 113]. The following description of MMI is a summary of that found in [75]. The MMI approach attempts to find the model parameters θ which minimize the conditional entropy $H_\theta(\mathbf{C}|\mathbf{X})$ of the random

variable $\mathbf{C}$ (class) given the random variable $\mathbf{X}$ (data), i.e.

$$H_\theta(\mathbf{C}|\mathbf{X}) = -\sum_{c,x} P(\mathbf{C} = c, \mathbf{X} = x) log P_\theta(\mathbf{C} = c|\mathbf{X} = x), \tag{3.1}$$

which represents the uncertainty of $\mathbf{C}$ given that we have observed $\mathbf{X}$. The entropy of a discrete random variable $\mathbf{C}$, which is a measure of the uncertainty of $\mathbf{C}$, is defined as

$$H_\theta(\mathbf{C}) = -\sum_{c} P(\mathbf{C} = c) log P_\theta(\mathbf{C} = c). \tag{3.2}$$

The information provided by $\mathbf{X}$ about $\mathbf{C}$ can then be defined as

$$I_\theta(\mathbf{C}; \mathbf{X}) = H_\theta(\mathbf{C}) - H_\theta(\mathbf{C}|\mathbf{X}). \tag{3.3}$$

Minimizing the conditional entropy, can therefore be accomplished by maximizing the mutual information between $\mathbf{C}$ and $\mathbf{X}$, $I_\theta(\mathbf{C}; \mathbf{X})$, i.e.

$$I_\theta(\mathbf{C}; \mathbf{X}) = \sum_{c,x} P(C = c, X = x) log \frac{P_\theta(C = c, X = x)}{P_\theta(C = c) P_\theta(X = x)}. \tag{3.4}$$

MMI maximizes the difference between $P_\theta(C = c, X = x)$ and $P_\theta(X = x) = \sum_c P_\theta(C = c, X = x)$. Unfortunately, maximizing the mutual information does not necessarily minimize the classification error.

Minimum classification error Juang and Katagiri [60] proposed a new formulation for the minimum classification error problem, together with a fundamental technique for designing classifiers that approach the objective of minimum classification error. The method was applied to multilayer neural networks, with significant improvements in performance over traditional training methods.

The minimum classification error training method, as introduced by Juang and Katagiri [60], has been used extensively in speech recognition. Applications thereof include training of neural networks [60, 104] and dynamic time warping (DTW) [64, 17, 76, 77] and hidden Markov models. The remainder of this section presents a concise literature survey of the usage of the MCE procedure to train HMM systems for speech recognition.

Chou *et al.* [23] introduced a segmental generalized probabilistic descent (GPD) training algorithm for HMM based speech recognizers using Viterbi decoding. Instead of using the forward-backward procedure, they proposed using the best state sequence obtained using the Viterbi algorithm. Instead of using a complicated constrained GPD algorithm, they apply segmental GPD to transformed HMM parameters, thereby ensuring that the HMM constraints are maintained. They reported results for both the E-set and TIDIGIT database. Significant improvements in phonetic classification from 76% to 88.3% were reported for the E-set problem when using a 10 state, 5 mixture HMM. Their results for the connected digit experiment (TIDIGITS) resulted in an improvement in continuous digit recognition rates from 98.7% to 98.8%. A more general and complete article was later published [59].

Chou *et al.* [24] later introduced a *minimum string error rate* training algorithm, based in the N-best string models. Here, the MCE criterion is applied at string level, with the goal of minimizing the string error rate in continuous and large vocabulary speech recognition tasks. The N most confusable strings are obtained by using the tree-trellis N-best search of Soong and Huang [106]. Their MCE algorithm was tested using the TIDIGIT database and the speaker independent portion of the DARPA naval resource management (RM) speech recognition task. An improvement in string error rate from 1.3% to 1.0% was reported for the TIDIGIT dataset, while a word error rate reduction of 17%-20% was observed when using the DARPA RM task.

McDermott [75] investigated the usage of string-level MCE. A second order optimization algorithm to minimize the string-level MCE criterion was described and found to be a reasonable alternative to the GPD algorithm. McDermott defined a more gen-

eral MCE loss function which attempted to represent finer grained differences between the correct and incorrect strings, however, no significant advantage was found as a result of using this new loss function. A second loss function, also designed to reflect phoneme/word accuracy was proposed, but not evaluated. The TIMIT dataset was used to compare string-level MCE and the baseline ML-trained HMM systems, the results of which are summarized in Table 2.5.

Kwon and Un [65] proposed a new method of finding discriminative state weights recursively using the MCE algorithm. They relax the HMM constraints on state-weights, such that the sum of the mixture weights for an HMM sum to the number of states. The mixture weights for an individual state can therefore sum to a value greater or smaller than one. This results in what could be called state-weighting, where certain states have higher weights than others. The MCE algorithm was then used to estimate the weights. Experimental results showed that recognizers with phoneme-based and word-based state-weights achieved a 20% and 50% decrease in word error rate respectively for isolated word recognition, and a 5% decrease in error rate for continuous speech recognition.

Other applications of MCE within a speech recognition framework include speaker adaptation [74], keyword spotting [109], speaker identification [105] and feature extraction [9, 8].

The implementation of MCE discussed in this thesis is based on the work of Chou *et al.* [23, 59, 24]. The work presented in this chapter has been summarized in our ICSLP 2000 article [89].

3.2 Minimum classification error training

The aim of minimum classification error (MCE) training is to correctly discriminate the observations of an HMM for best recognition results and not to fit the distributions

to the data.

This section will briefly describe and discuss the MCE algorithm; for a more detailed discussion the reader is referred to the original work of Juang and Katagiri [60] and Juang *et al.* [59].

3.2.1 Bayes risk

The following is a brief introduction to Bayes risk and Bayes decisions; for a more detailed text on this subject the reader is referred to the books of DeGroot [27] and Duda and Hart [40]. The optimal choice of answer for an inference problem is a $\theta \in \Theta$ which maximizes the expected utility [7],

$$\int_{\mathcal{X}} u(C(\mathbf{x};\theta)|\mathbf{x})p(\mathbf{x})d\mathbf{x}, \tag{3.5}$$

where $C(\mathbf{x};\theta)$ is the classifiers decision for the observation $\mathbf{x}$, u is a function attaching utilities to each consequence of a decision and $\mathcal{X}$ is the set containing all possible observations. Alternatively, we could work with a loss function $l(C(\mathbf{x})|\mathbf{x})$, where

$$l(C(\mathbf{x})|\mathbf{x}) = f(x) - u(C(\mathbf{x})|\mathbf{x}), \tag{3.6}$$

where f is an arbitrary, fixed function. The optimal solution is then the value of θ which maximizes the expected loss,

$$\int_{\mathcal{X}} l(C(\mathbf{x};\theta)|\mathbf{x})p(\mathbf{x})d\mathbf{x}. \tag{3.7}$$

The conditional loss $l(C_i|\mathbf{x})$, or the risk of classifying the observation $\mathbf{x}$ into class i can be defined as

$$l(C_i|\mathbf{x}) = \sum_{j=1}^{N} \lambda_{ij} P(C_j|\mathbf{x}), \tag{3.8}$$

where $P(C_j|\mathbf{x})$ is the *a-posteriori* probability of choosing the class j given the data $\mathbf{x}$, which can be easily obtained using Bayes' theorem if the class-conditional densities of the data are known. The value λ_{ij} is the cost of classifying a class i observation as class j. Typically, the costs used in the loss function are chosen to be the zero-one loss function or

$$\lambda_{ij} = \begin{cases} 0 & i = j \\ 1 & i \neq j, \end{cases} \tag{3.9}$$

which associates zero cost with correct classifications and unity cost for incorrect classifications. For this special case the conditional loss becomes

$$\begin{aligned} l(C_i|\mathbf{x}) &= \sum_{i \neq j} P(C_j|\mathbf{x}) \\ &= 1 - P(C_i|\mathbf{x}), \end{aligned} \tag{3.10}$$

which is the probability of an error in classification and the minimum risk classifier is the classifier which delivers the minimum classification error. The classifier which achieves minimum loss uses the following decision rule

$$C(\mathbf{x}) = C_i \quad \text{where} \quad i = \underset{j}{argmax} P(C_j|\mathbf{x}). \tag{3.11}$$

3.2.2 Optimization criterion

The error rate for a finite data set is a piecewise constant function of the classifier parameter θ and therefore a poor candidate for optimization using a numerical search. It is therefore necessary to define an optimization criterion which provides a reasonable estimate of the error probability.

For HMMs, the decision rule can be stated as follows,

$$C(\mathbf{x}) = C_i \quad \text{where} \quad i = \underset{j}{argmax} P_j(\mathbf{O}|\theta_j), \tag{3.12}$$

where $P_j(\mathbf{O}|\theta_j)$ is the log-likelihood of the input utterance or observation sequence ($\mathbf{O} = \{\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_n\}$) for the j-th model.

It is therefore necessary to express the operational decision rule (Eq. (3.12)) in a functional form. A class misclassification measure which attempts to emulate the decision rule is therefore defined [59],

$$d(\mathbf{O}) = -\ln[P_i(\mathbf{O}|\theta_i)] + \ln[\frac{1}{N}\sum_{j,j\neq i}^{N} e^{\ln[P_j(\mathbf{O}|\theta_j)]\eta}]^{1/\eta}, \tag{3.13}$$

where η is a positive number, and N is the number of N-best incorrect classes which are used in the misclassification measure. The value of η influences the behaviour of the right-hand term in Eq. (3.13); for $\eta = \infty$ the term becomes $max_{j,j\neq i} P_j(\mathbf{O}|\theta_j)$. A large η will result in the misclassification measure only incorporating the closest incorrect class, whereas a small η will include contributions from all of the incorrect classes. The misclassification measure is a continuous function of the classifier parameters. A value of $d(O) > 0$ implies a misclassification and $d(O) < 0$ a correct decision (for $\eta = \infty$).

The complete loss function is then defined in terms of the misclassification measure using a zero-one loss function,

$$l_i(\mathbf{O};\theta) = l(d(\mathbf{O})). \tag{3.14}$$

It is worth noting, that using $\eta = 1$ and loss function $l(d) = d$ [101] results in an optimization criterion very close to that of MMI (Eq. (3.4)). The zero-one loss function can be any continuous zero-one function, but is typically the following sigmoid function

$$l(d) = \frac{1}{1 + e^{-\gamma d + \lambda}}, \tag{3.15}$$

where λ is typically set to zero (or slightly smaller than zero). A loss value less than 0.5 indicates a misclassification has occurred (assuming $\lambda = 0$).

For a given observation, the classifier performance can be written as

$$l(\mathbf{O};\theta) = \sum_{i=1}^{N} l_i(\mathbf{O};\theta) I(X \in C_i), \tag{3.16}$$

where $I(\cdot)$ is the indicator function.

3.2.3 Optimization methods

As mentioned in Section 3.2.1, the optimal solution for an inference problem is that which minimizes the expected loss. For a classification problem involving N classes, the expected loss is defined as

$$L(\theta) = E_{\mathbf{O}}[l(\mathbf{O};\theta)] = \sum_{i=1}^{N} \int_{\mathbf{O}\in C_i} l_i(\mathbf{O};\theta) p(\mathbf{O}) d\mathbf{O}. \tag{3.17}$$

The generalized probabilistic descent (GPD) algorithm [60] is used to minimize the expected loss. The GPD algorithm is given by:

$$\theta_{t+1} = \theta_t - \epsilon_t U_t \nabla l(\mathbf{O}; \theta)|_{\theta=\theta_t} \tag{3.18}$$

where U_t is a positive definite matrix, ϵ_t is the learning rate or step size of the adaptation, and θ_t is the model parameters at time step t. It can be shown [60] that the expected loss converges to a local minimum when using the GPD algorithm and the following conditions are satisfied

$$\sum_{t=1}^{\infty} \epsilon_t \to \infty, \quad \text{and} \tag{3.19}$$

$$\sum_{t=1}^{\infty} \epsilon_t^2 < \infty. \tag{3.20}$$

Using the class conditional likelihood function $P_i(\mathbf{O}|\theta_i)$, i.e.

$$P_i(\mathbf{O}|\theta_i) = \sum_{\text{all } \mathbf{q}} P_i(\mathbf{O}, \mathbf{q}|\theta_i), \tag{3.21}$$

where $\mathbf{q}$ is a given state sequence, requires the use of the relatively computationally expensive forward-backward procedure [90, p. 334]. We can, however, also use the maximum of the joint observation-state probability, i.e.

$$P_i(\mathbf{O}|\theta_i) \approx \max_{\mathbf{q}} [P_i(\mathbf{O}, \mathbf{q}|\theta_i)]. \tag{3.22}$$

Since the best state sequence is segmented using the Viterbi algorithm and this segmented sequence is used in the calculation of $P_i(\mathbf{O}|\theta_i)$, this instance of the MCE algorithm based on Eq. (3.22) is often referred to as segmental MCE.

3.2.4 Parameter transformation

The GPD algorithm is an unconstrained optimization technique and given that certain constraints must be maintained for HMMs, some modifications are required. Instead of using a complicated constrained GPD algorithm, Chou *et al.* [23] applied GPD to transformed HMM parameters. The parameter transformations ensure that there are no constraints in the transformed space where the updates occur. The following HMM constraints should be maintained,

1. $\sum_j a_{ij} = 1$ and $a_{ij} \geq 0$,
2. $\sum_k c_{jk} = 1$ and $c_{jk} \geq 0$, and
3. $\sigma_{jkl} \geq 0$.

The parameter transformations given in Table 3.1 are therefore used before and after parameter adaptation (Eq. 3.18).

Table 3.1: Parameter transformations used in MCE.

Parameter	Transformed parameter	Forward transform	Reverse transform	
a_{ij}	$\tilde{a}_{ij}$	$\tilde{a}_{ij} = \ln(a_{ij})$	$a_{ij} = \frac{e^{\tilde{a}_{ij}}}{\sum_j e^{\tilde{a}_{ij}}}$	Transition probabilities
c_{jk}	$\tilde{c}_{jk}$	$\tilde{c}_{jk} = \ln(c_{jk})$	$c_{jk} = \frac{e^{\tilde{c}_{jk}}}{\sum_k e^{\tilde{c}_{jk}}}$	Mixture weights
μ_{jkl}	$\tilde{\mu}_{jkl}$	$\tilde{\mu}_{jkl} = \frac{\mu_{jkl}}{\sigma_{jkl}}$	$\mu_{jkl} = \sigma_{jkl}\tilde{\mu}_{jkl}$	Gaussian mean
σ_{jkl}	$\tilde{\sigma}_{jkl}$	$\tilde{\sigma}_{jkl} = \ln(\sigma_{jkl})$	$\sigma_{jkl} = e^{\tilde{\sigma}_{jkl}}$	Gaussian std. dev.

The sensitivity of the mean parameter update is determined by the size of the associated variance. The positive definite matrix U_t should therefore be chosen carefully, so as to compensate for this sensitivity. Chou *et al.* [23] used a diagonal matrix, where the diagonal elements were equal to the variances (for the mean parameter update). This is equivalent to using the mean parameter transformation in Table 3.1 with the matrix

U_t equal to the identity matrix and has been used throughout. Under these conditions, GPD reverts to the simpler gradient descent algorithm (for all parameters), where

$$\theta_{t+1} = \theta_t - \epsilon_t \nabla l(\mathbf{O}; \theta)|_{\theta=\theta_t}. \tag{3.23}$$

3.2.5 Parameter adaptation

It can be easily verified that the partial derivative of $l_i(d_i)$ (Eq. (3.15)) with respect to the misclassification measure (d_i) is

$$\frac{\partial l_i}{\partial d_i} = \gamma d_i(1 - d_i). \tag{3.24}$$

Calculation of the parameter update defined in Eq. (3.18) or (3.23) requires the gradient function $\nabla l(\mathbf{O}; \theta)$ or $\frac{\partial l(\mathbf{O};\theta)}{\partial \theta}$. In what follows, the gradients and parameter updates for the four parameter types (mean, variance, mixture weights and transition probabilities) are derived.

Gaussian mixture means

According to Eq. (3.23) the parameter update for the mean vectors of HMM i is as follows:

$$\tilde{\mu}^{(i)}_{jkl}(n+1) = \tilde{\mu}^{(i)}_{jkl}(n) - \epsilon \frac{\partial l_i(\mathbf{O}_n; \theta_n)}{\partial \tilde{\mu}^{(i)}_{jkl}}. \tag{3.25}$$

The partial derivative of $l_i(\mathbf{O}_n; \theta_n)$ with respect to $\tilde{\mu}^{(i)}_{jkl}$ can be obtained using the chain rule

$$\frac{\partial l_i(\mathbf{O}_n;\theta_n)}{\partial \tilde{\mu}_{jkl}^{(i)}} = \frac{\partial l_i}{\partial d_i}\frac{\partial d_i}{\partial \tilde{\mu}_{jkl}^{(i)}}, \tag{3.26}$$

where $\frac{\partial l_i}{\partial d_i}$ can be calculated using Eq. (3.24), and from Eq. (3.13) we get

$$\frac{\partial d_i}{\partial \tilde{\mu}_{jkl}^{(i)}} = \begin{cases} -\frac{\partial f(\mathbf{O};\theta_n)}{\partial \tilde{\mu}_{jkl}^{(i)}} & \text{correct class} \\ \frac{e^{\eta f_i(\mathbf{O};\theta_n)}}{\sum_{j,j\neq i} e^{\eta f_j(\mathbf{O};\theta_n)}}\frac{\partial f(\mathbf{O};\theta_n)}{\partial \tilde{\mu}_{jkl}^{(i)}} & \text{incorrect class,} \end{cases} \tag{3.27}$$

where $f_i(\mathbf{O};\theta_n) = \ln P_i(\mathbf{O}|\theta_n)$. From Eq. (2.9) we have

$$\frac{\partial f(\mathbf{O};\theta_n)}{\partial \tilde{\mu}_{jkl}^{(i)}} = \sum_{t=1}^{T} \delta(\bar{q}_t - j)\frac{\partial \ln(b_j^{(i)}(\mathbf{o}_t))}{\partial \tilde{\mu}_{jkl}^{(i)}}, \tag{3.28}$$

where $\delta()$ is the Kronecker delta function, and

$$\begin{aligned} \frac{\partial \ln(b_j^{(i)}(\mathbf{o}_t))}{\partial \tilde{\mu}_{jkl}^{(i)}} =& c_{jk}^{(i)}(2\pi)^{-d/2}|\Sigma_{jk}^{(i)}|^{-1/2}(b_j^{(i)}(\mathbf{o}_t))^{-1}\left(\frac{\mathbf{o}_{tl}}{\sigma_{jkl}^{(i)}} - \tilde{\mu}_{jkl}^{(i)}\right) \\ & e^{-\frac{1}{2}\sum_{l=1}^{D}\left(\frac{o_{tl}}{\sigma_{jkl}^{(i)}} - \tilde{\mu}_{jkl}^{(i)}\right)^2} \\ =& \frac{c_{jk}\mathcal{N}(\mathbf{o}_t;\mu_{jk},\Sigma_{jk})}{b_j^{(i)}(\mathbf{o}_t)}\cdot\left(\frac{o_{tl}-\mu_{jkl}}{\sigma_{jkl}}\right). \end{aligned} \tag{3.29}$$

Having updated the transformed mean using Eq. (3.25), the correct mean can be found using the inverse transformation $\mu_{jkl}^{(i)}(n+1) = \tilde{\mu}_{jkl}^{(i)}(n)\sigma_{jkl}^{(i)}(n+1)$.

Gaussian mixture variance

The Gaussian mixture variance update is

$$\tilde{\sigma}_{jkl}^{(i)}(n+1) = \tilde{\sigma}_{jkl}^{(i)}(n) - \epsilon \frac{\partial l_i(\mathbf{O}_n;\theta_n)}{\partial \tilde{\sigma}_{jkl}^{(i)}}, \tag{3.30}$$

where

$$\frac{\partial l_i(\mathbf{O}_n;\theta_n)}{\partial \tilde{\sigma}_{jkl}^{(i)}} = \frac{\partial l_i}{\partial d_i}\frac{\partial d_i}{\partial \tilde{\sigma}_{jkl}^{(i)}}. \tag{3.31}$$

The partial derivative $\frac{\partial l_i}{\partial d_i}$ is obtained using Eq. (3.24) and again from Eq. (3.13) we have

$$\frac{\partial d_i}{\partial \tilde{\sigma}_{jkl}^{(i)}} = \begin{cases} -\frac{\partial f(\mathbf{O};\theta_n)}{\partial \tilde{\sigma}_{jkl}^{(i)}} & \text{correct class} \\ -\frac{e^{\eta f_i(\mathbf{O};\theta_n)}}{\sum_{j,j\neq i} e^{\eta f_j(\mathbf{O};\theta_n)}}\frac{\partial f(\mathbf{O};\theta_n)}{\partial \tilde{\sigma}_{jkl}^{(i)}} & \text{incorrect class,} \end{cases} \tag{3.32}$$

where

$$\frac{\partial f(\mathbf{O};\theta_n)}{\partial \tilde{\sigma}_{jkl}^{(i)}} = \sum_{t=1}^{T} \delta(\bar{q}_t - j)\frac{\partial \ln(b_j^{(i)}(\mathbf{o}_t))}{\partial \tilde{\sigma}_{jkl}^{(i)}} \tag{3.33}$$

$$\frac{\partial \ln(b_j^{(i)}(\mathbf{o}_t))}{\partial \tilde{\sigma}_{jkl}^{(i)}} = \frac{c_{jk}\mathcal{N}(\mathbf{o}_t;\mu_{jk},\Sigma_{jk})}{b_j^{(i)}(\mathbf{o}_t)} \cdot \left[\left(\frac{o_{tl} - \mu_{jkl}^{(i)}}{\sigma_{jkl}^{(i)}}\right)^2 - 1\right]. \tag{3.34}$$

Finally, we use

$$\sigma_{jkl}^{(i)}(n+1) = e^{\tilde{\sigma}_{jkl}(n+1)}. \tag{3.35}$$

Gaussian mixture weights

The MCE update for the Gaussian mixture weights is

$$\tilde{c}_{jk}^{(i)}(n+1) = \tilde{c}_{jk}^{(i)}(n) - \epsilon \frac{\partial l_i(\mathbf{O}_n;\theta_n)}{\partial \tilde{c}_{jk}^{(i)}}, \tag{3.36}$$

where

$$\frac{\partial l_i(\mathbf{O}_n;\theta_n)}{\partial \tilde{c}_{jk}^{(i)}} = \frac{\partial l_i}{\partial d_i}\frac{\partial d_i}{\partial \tilde{c}_{jk}^{(i)}}. \tag{3.37}$$

From Eq. (3.13), we have

$$\frac{\partial d_i}{\partial \tilde{c}_{jk}^{(i)}} = \begin{cases} -\frac{\partial f(\mathbf{O};\theta_n)}{\partial \tilde{c}_{jk}^{(i)}} & \text{correct class} \\ \frac{e^{\eta f_i(\mathbf{O};\theta_n)}}{\sum_{j,j\neq i} e^{\eta f_j(\mathbf{O};\theta_n)}} \frac{\partial f(\mathbf{O};\theta_n)}{\partial \tilde{c}_{jk}^{(i)}} & \text{incorrect class,} \end{cases} \tag{3.38}$$

where

$$\frac{\partial f(\mathbf{O};\theta_n)}{\partial \tilde{c}_{jk}^{(i)}} = \sum_{t=1}^{T} \delta(\bar{q}_t - j) \frac{\partial \ln(b_j^{(i)}(\mathbf{o}_t))}{\partial \tilde{c}_{jk}^{(i)}}, \tag{3.39}$$

$$\frac{\partial \ln(b_j^{(i)}(\mathbf{o}_t))}{\partial \tilde{c}_{jk}^{(i)}} = c_{jk}^{(i)} \left[\frac{\mathcal{N}(\mathbf{o}_t;\mu_{jk},\Sigma_{jk})}{b_j^{(i)}(\mathbf{o}_t)} - 1 \right]. \tag{3.40}$$

Finally, we use

$$c_{jk}^{(i)}(n+1) = \frac{e^{\tilde{c}_{jk}(n+1)}}{\sum_k e^{\tilde{c}_{jk}(n+1)}}. \tag{3.41}$$

Transition probabilities

The update for the transition probabilities is

$$\tilde{a}_{ij}^{(i)}(n+1) = \tilde{a}_{ij}^{(i)}(n) - \epsilon \frac{\partial l_i(\mathbf{O}_n;\theta_n)}{\partial \tilde{a}_{ij}^{(i)}}, \tag{3.42}$$

where

$$\frac{\partial l_i(\mathbf{O}_n;\theta_n)}{\partial \tilde{a}_{ij}^{(i)}} = \frac{\partial l_i}{\partial d_i}\frac{\partial d_i}{\partial \tilde{a}_{ij}^{(i)}}. \tag{3.43}$$

From Eq. (3.13), we get

$$\frac{\partial d_i}{\partial \tilde{a}_{ij}^{(i)}} = \begin{cases} -\frac{\partial f(\mathbf{O};\theta_n)}{\partial \tilde{a}_{ij}^{(i)}} & \text{correct class} \\ \frac{e^{\eta f_i(\mathbf{O};\theta_n)}}{\sum_{j,j\neq i} e^{\eta f_j(\mathbf{O};\theta_n)}} \frac{\partial f(\mathbf{O};\theta_n)}{\partial \tilde{a}_{ij}^{(i)}} & \text{incorrect class.} \end{cases} \tag{3.44}$$

From Eq. 2.9,

$$\frac{\partial f(\mathbf{O};\theta_n)}{\partial \tilde{a}_{ij}^{(i)}} = \sum_{t=1}^{T}\sum_{s} \delta(\bar{q}_{t-1} - i)\delta(\bar{q}_t - s)\Big[\delta(j-s) - a_{ij}\Big]. \tag{3.45}$$

Finally, the reverse transformation is applied,

$$a_{ij}^{(i)}(n+1) = \frac{e^{\tilde{a}_{ij}(n+1)}}{\sum_k e^{\tilde{a}_{ij}(n+1)}}. \tag{3.46}$$

3.3 Embedded MCE

In the derivation of the MCE algorithm in the previous sections, it was assumed that the whole training observation $\mathbf{O}$ must be one of N classes. MCE can, however, be applied at the level of various speech units, such as phonemes, words and sentences. Here, a search is used to automatically segment (state and HMM alignment) and label

the utterance for usage within the MCE framework. More specifically, the correct alignment and the N-best incorrect alignments are required, for which an N-best search is used.

The application of MCE to strings of phoneme or word units is known as string-level MCE or embedded MCE [59, 78, 75, 24]. Here, the observation $\mathbf{O}$ is a concatenated string of observations belonging to different classes. In this situation, MCE is used to minimize the string error rate and not the individual phoneme or word error rates. Note that although the phoneme or word error rates are not directly minimized, minimizing the string error rate will tend to decrease the phoneme and word error rates.

The MCE procedure remains the same, except that in this case the correct string and N-best strings are required, as opposed to the single correct and N-best incorrect acoustic units. The N-best search proposed by Soong and Huang [106] to generate the N-best alignments and subsequently used by Chen and Soong [18, 19] and McDermott [75] for string-level MCE was implemented and used in our work.

Figure 3.1 shows a state occupancy diagram for a simple example where embedded MCE is used. Here, the observation is of the word "boot", containing the phonetic units b, uw and t, with silence on either side. The state occupancy diagram for the correct string is given at the top, with the state occupancy diagram of the most confusable incorrect string below it. Let us assume, for the purpose of this example, that we are only working with the correct string and the single best incorrect string. State occupation is indicated with thick black or grey lines. A grey line is used to indicate when an error in state occupancy occurs.

When state occupancy is correct for both the correct and best incorrect string (black lines in incorrect string state occupancy in Figure 3.1), the gradient of Eq. (3.13) with respect to the parameters will be zero (gradients due to correct and incorrect strings are equal, but of opposite sign). The models d and aw in the incorrect string are a substitution and insertion respectively and the state alignments associated with

these units are therefore incorrect. Although the units *uw* and *t* are in the correct position, the state alignment is not correct and the state alignments displayed in grey will result in the associated state being penalized. Therefore, incorrect state occupancy will result in the model associated with the correct alignment being reinforced and the model associated with the incorrect class penalized according to the update functions defined earlier.

3.4 Discussion and Experiments

The following sections will discuss and experimentally validate a number of issues related to the use and implementation of the MCE algorithm. They are:

- batch-mode versus online optimization,
- smoothness of the loss function,
- the need for a zero-one loss function, and
- overtraining in MCE and several solutions.

The TIMIT dataset, described in Section 2.4.1, were used for this purpose. Two training sets were used, namely the standard TIMIT training set T and the small gender independent training set T_S, which were described in Section 2.4.1. The relevant training and testing set details are reproduced in Table 3.2 for convenience. The basic configuration of the system is as described in Section 2.1. A 3 state, 5 mixture HMM is used to model each of the 39 phonetic units in the TIMIT dataset.

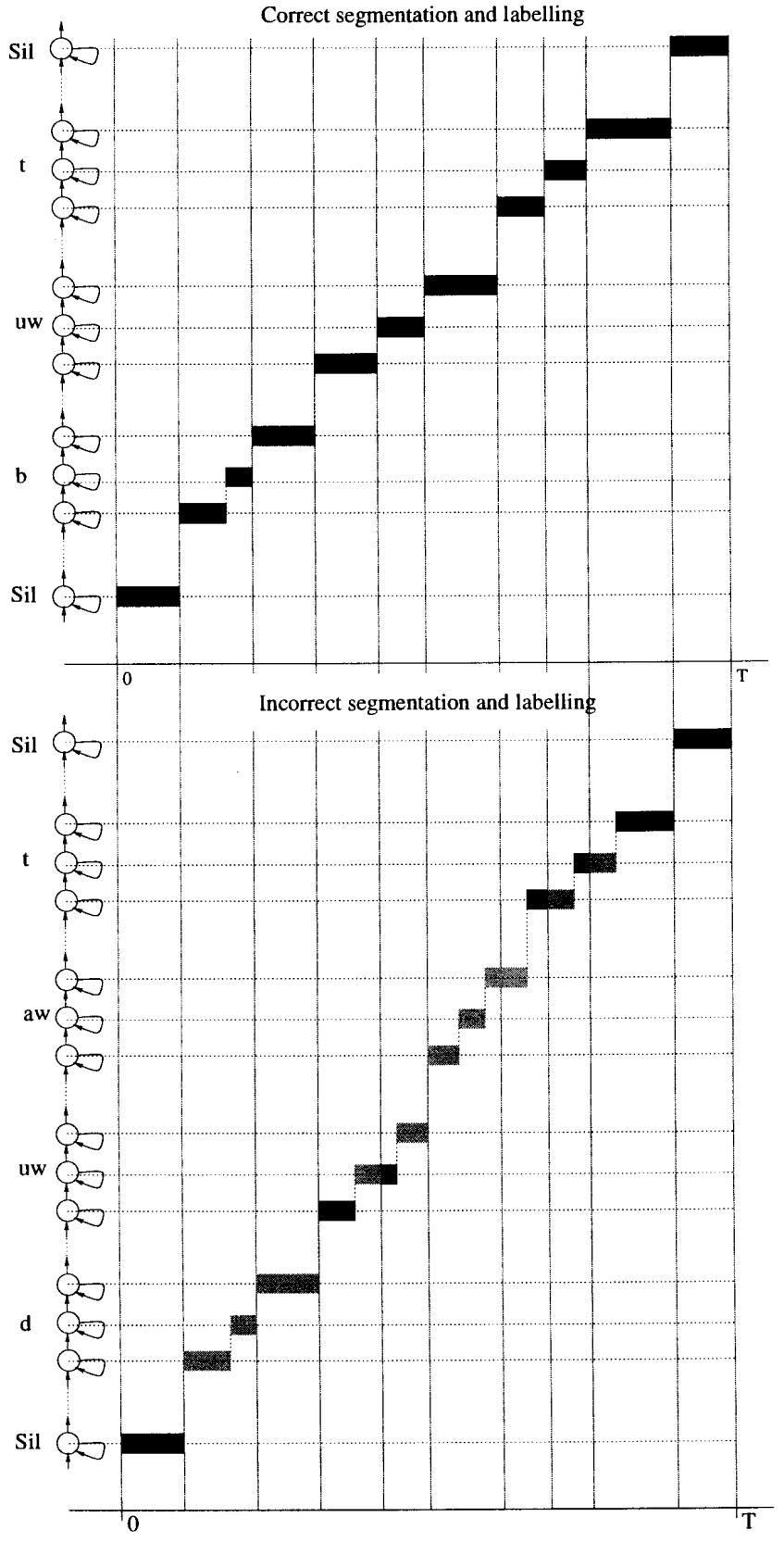


Figure 3.1: State occupancy diagram of the correct and an incorrect string

Table 3.2: Description of TIMIT training and testing sets used.

Description	Label	Number of speakers			Number of sentences	Duration (minutes)
		Male	Female	Total		
Training sets:						
Full (standard)	T	326	136	462	4620	236.5
Small (gender indep.)	T_S	16	16	32	320	16.1
Testing set (standard)		112	56	168	1680	86.4

3.4.1 Batch-mode versus online optimization

Online (stochastic) optimization algorithms use a single training example to determine the parameter update (Eq. (3.18) in this case) for each training example. The deterministic (batch-mode) descent algorithm computes the combined gradient for all the training examples in the training set, which is then used in the parameter update.

The online descent algorithm can perform better in situations where there is redundancy in the training data. The number of passes through the data to find a local minimum is therefore often less for online than for batch-mode optimization. It is, however, necessary to randomly select examples without replacement when using the online algorithm. If not done, oscillatory and non-optimal behaviour might be observed in the optimization process, as a result of data set structure.

Figure 3.2 presents the performance of online and batch-mode MCE using the standard TIMIT training and testing sets. The learning rates for both algorithms were set just below the point at which they became unstable. The online descent algorithm is considerably faster in terms of training time, as opposed to deterministic (batch) gradient descent. Maximum training and testing set performance is better when using online optimization – batch-mode optimization has found a less optimal local minimum. Online descent is therefore preferred and is used throughout.

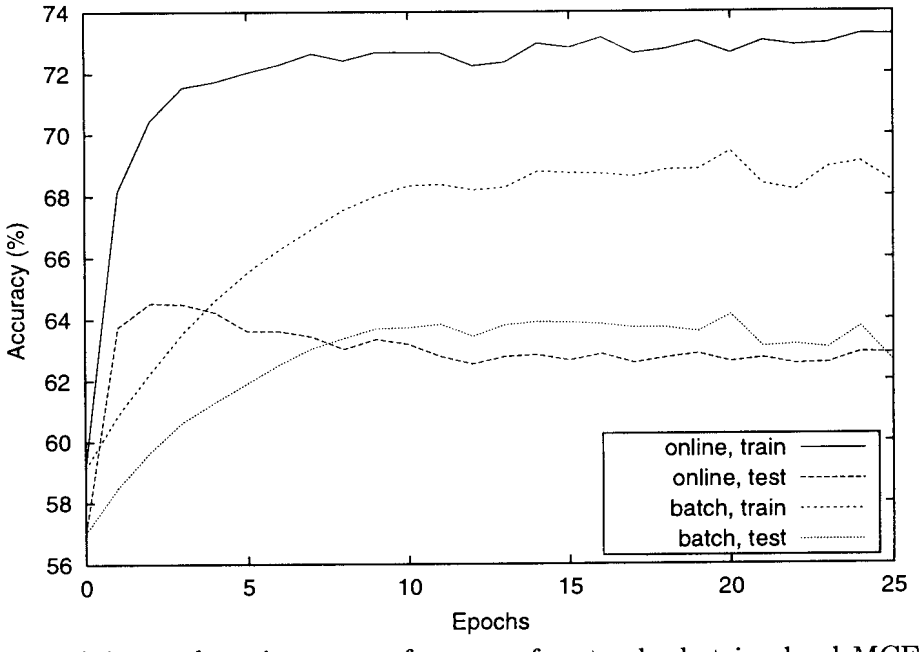


Figure 3.2: Training and testing set performance for standard string-level MCE using the batch and online optimization algorithms

3.4.2 Smoothness of the loss function

The smoothness of the zero-one loss function (Eq. (3.15)) is somewhat critical to the performance of the MCE algorithm. A zero-one loss function which is reasonably sharp, limits the effect of individual examples on the loss and allows only those examples on or near the decision boundaries to affect the parameter update. Outliers will therefore effectively not be included in the gradient calculation, thereby promoting robustness. A smoother loss function, is less likely to be caught in local minima. This effect will be further discussed in Section 3.4.3.

McDermott [75] stated that too much smoothing would result in a discrepancy between the function being optimized and the target, minimum classification error. I, however, do not believe that this is the case, as an extremely smooth sigmoid loss function is linear in the region of interest and so the misclassification measure, which is designed to emulate the decision rule, is therefore directly minimized. It is, however, true that a discrepancy results between the function being optimized and the target when the

zero-one loss function is relatively sharp. Here, the function being optimized can be interpreted as "*minimum classification error for only those utterances which have misclassification measures close to zero*".

Figure 3.3 shows a histogram of the misclassification measure values for the utterances in the TIMIT training set, when using a 3 state 5 mixture HMM. All of the misclassification measure values are greater than zero, which implies that none of the utterances were correctly classified (only strictly true when $\eta = \infty$). The sigmoid loss function (Eq. (3.15)) output for $\gamma = 0.1$, $\gamma = 0.01$ and $\gamma = 0.001$ are also shown. It is important that the derivative of the loss function must be significant for a large part of the training data. The loss function with $\gamma = 0.1$ is too sharp and only has a reasonable derivative for small misclassification measure values, which will result in only a few of the utterances influencing the update. One would, therefore, not expect the MCE algorithm to work well for $\gamma = 0.1$. The loss function using $\gamma = 0.01$ is mostly linear for a reasonable part of the training data, and has a significant derivative for a large part of the training data. Using $\gamma = 0.001$ results in a loss function which is effectively linear for all of the training utterances. The optimal value for γ will therefore, probably lie somewhere around 0.01.

Figure 3.4 shows the TIMIT test set accuracy versus the sigmoid loss function parameter γ for both the full training set T and the small training set T_S [1]. As expected, $\gamma = 0.1$ results in a sigmoid function which is too sharp and the accuracy attained (58.3% for T) is therefore considerably worse than that attained using smaller values of γ. Peak accuracy of 64.7% results when $\gamma = 0.01$ is used for the training set T. The MCE algorithm is considerably less effective when the small training set is used (T_S), resulting in a peak testing set performance of only 53.8%.

[1] Note that phone recognition accuracy results (Eq. (2.26)) are reported throughout this chapter

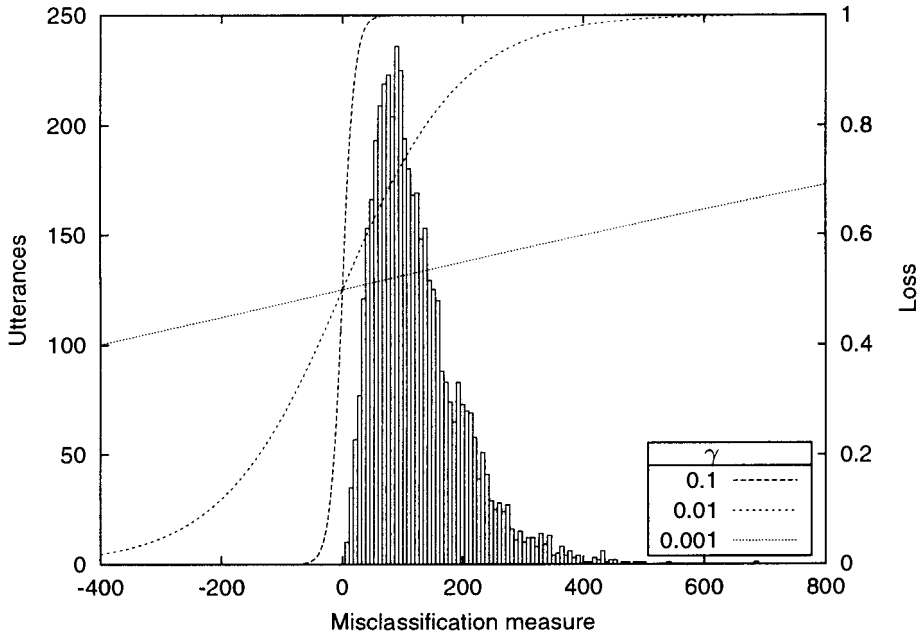


Figure 3.3: Histogram of the misclassification measure before embedded MCE training for the TIMIT training set. The sigmoid loss functions with $\gamma = 0.1$, $\gamma = 0.01$ and $\gamma = 0.001$ are plotted.

3.4.3 Need for a zero-one loss function

The question must be asked, "Why is a smoothed zero-one loss function needed?". The main reason why one would introduce a sigmoid function is to promote stability in the training process and to ensure that the resultant gradient function (and therefore the parameter update) is finite and continuous.

Figure 3.5 shows an example of a simple two class problem where a sigmoid loss function could result in the best minimum not being attained. Here we assume that the two classes can each be modeled by a single Gaussian distribution. Let us also assume that we know the variances are fixed and equal to one. The ML estimates of the means are (0.02, 4.23) for +'s and (0.25, 0.09) for ×'s. The theoretical minimum error classification boundary is then as shown by line (a) in Figure 3.5.

If the sigmoid function is quite sharp, then the derivative for the "outliers" in the

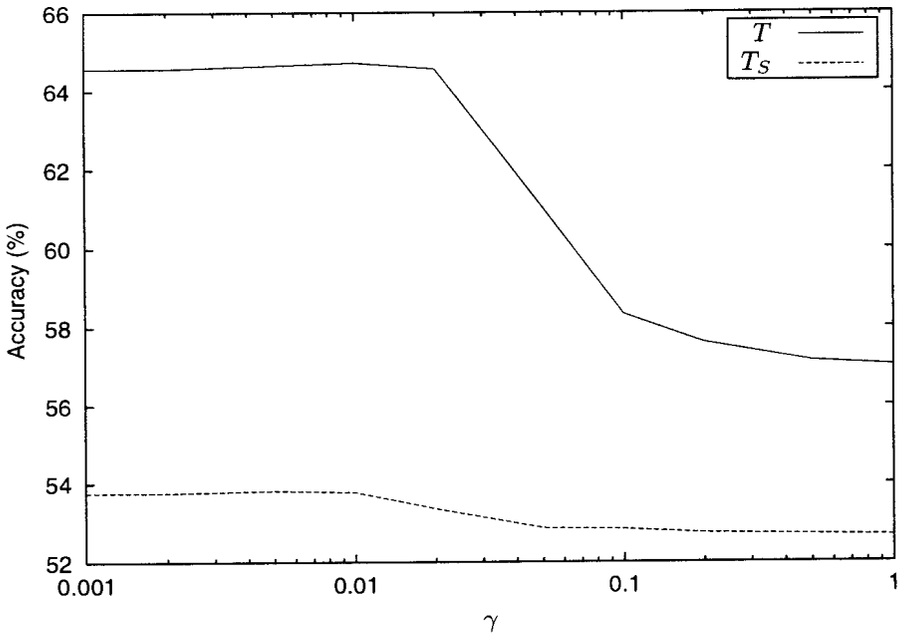


Figure 3.4: TIMIT test set accuracy versus γ for the training set T and T_S.

top-right corner will be close to zero and the class boundary (b), which has a lower classification error, will not be reached using MCE. This situation occurs because the true distributions are not Gaussian as we assumed they were. It is, therefore, unfortunate that by using a (relatively sharp) zero-one loss function we are introducing additional local minima. It is these local minima which can also in certain situations save the algorithm from overtraining. This could be the case if there is excessive noise in the training set, or if the data has been poorly labelled. Whether decision boundary (b) is truly better than (a) can only be determined by using an independent test set.

The results of the previous section (3.4.2), where small values of γ resulted in optimal performance, support the above discussion. Table 3.3 gives the results for the MCE algorithm with and without using a sigmoid loss function with $\gamma = 0.01$. There is no significant difference between the two algorithms, with the 0.1% difference for the large training set T due more to rounding than any algorithmic difference.

This result is important for modifications and algorithms proposed later in this and other chapters, where the algorithm or modification cannot be mathematically justified

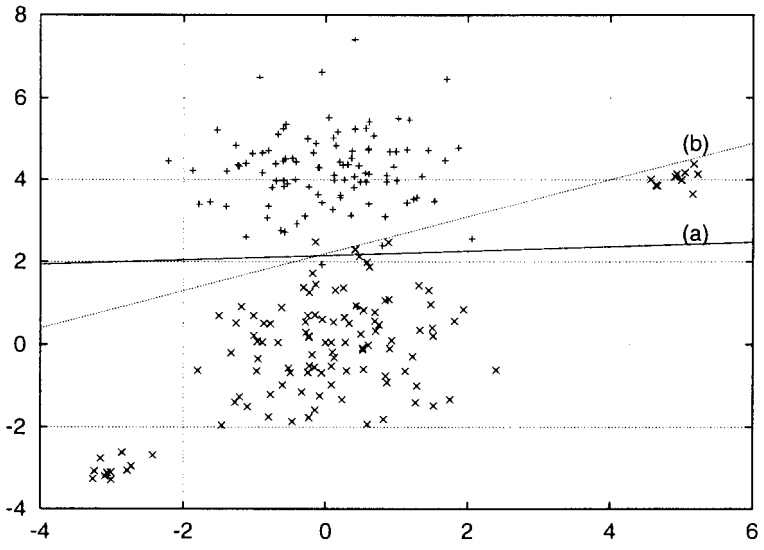


Figure 3.5: Example of where a sharp sigmoid does not shift the ML class boundary (a) to boundary (b), with lower classification error.

Table 3.3: Comparison of MCE accuracy results when using a sigmoid loss function and no sigmoid loss function

Training set	sigmoid $\gamma = 0.01$	no sigmoid
T	64.7	64.6
T_S	53.8	53.8

when using a sigmoid loss function. If there is a concern over the stability of the algorithm, a sigmoid loss function with a small value of γ can be used, so that the loss function is linear for a reasonable part of the training data. We can then ignore the sigmoid loss function when incorporating such modifications in the loss function or optimization criterion.

3.4.4 Overtraining in MCE

Minimum classification error (MCE) training is somewhat prone to overspecialization. This section investigates various techniques which improve performance and general-

ization of the MCE algorithm.

In Section 2.2 overtraining was discussed from a model complexity perspective. Here, it is the parameter estimation algorithm which results in further overspecialization. It is also an unfortunate tendency of MCE and other discriminative training algorithms to result in a decrease in testing set performance after maximum performance has been attained (in terms of training time).

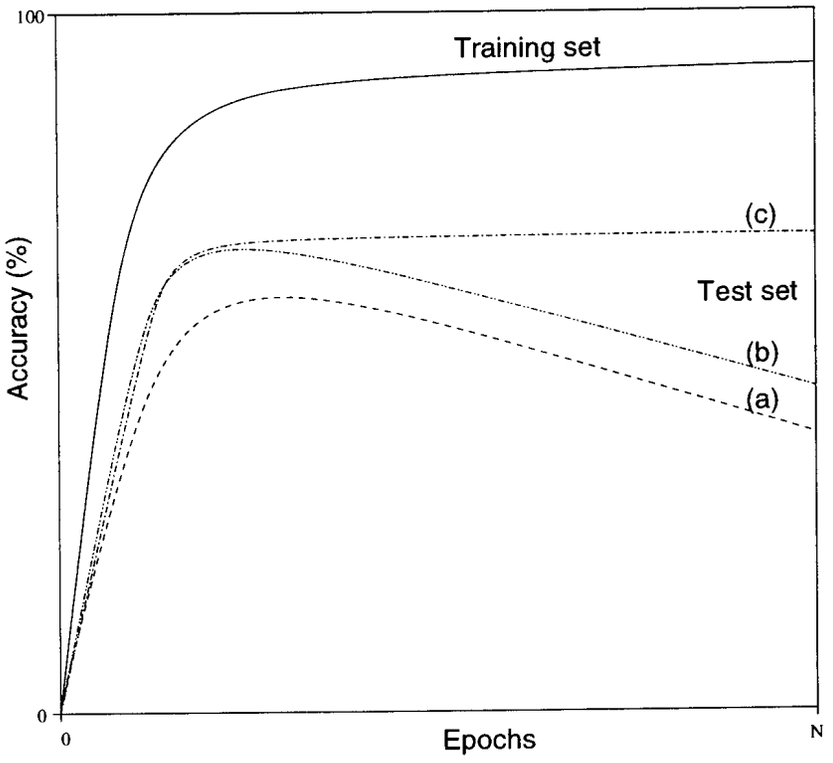


Figure 3.6: Comparison of typical (a) and preferred (b and c) testing set performance

Figure 3.6 shows the typical form of results obtained for the training and testing sets (a). Limiting specialization of the classifier would result in reducing the difference between training and testing set error rates. We would, for example, prefer result (b) in Figure 3.6 to result (a). We also wish to limit the degradation in performance after maximum performance has been reached. An algorithm which has the characteristics of (c) in Figure 3.6 would be advantageous in that a cross-validation set would not be required to choose the best model in an unbiased way.

Regularization techniques are often used to improve generalization. In regularization, a penalty term $F(\theta)$ which is called a regularizer is added to the original objective function, creating a new objective function, i.e.

$$\tilde{l}(\theta; \mathbf{O}) = l(\theta; \mathbf{O}) + \zeta F(\theta). \tag{3.47}$$

The regularizer conveys *a-priori* knowledge about the process which is to be learned. Shimodaira *et al.* [104] presented a method to prevent over-fitting and improve the generalization performance of the MCE algorithm when applied to neural networks. A simple version of the Tikhonov regularizer [11] was used for this purpose in [104]. This regularizer requires that the second order derivative of the likelihood with respect to the model parameters be computed. That is, however, not a simple task for hidden Markov models and was not incorporated in our work. The following sections discuss the two approaches that were followed to reduce overtraining.

Penalizing large variances

It can be expected that overspecialization would result in the variances of certain of the Gaussian mixtures becoming very small. To reduce overfitting, a penalty term proportional to the sum of the square (or power) of the inverse of the variances (precisions) of the Gaussians of the HMM states is therefore proposed. This is as expressed in Eq. (3.48), and is added to the loss function of MCE (Eq. (3.15)).

$$\alpha \sum_{\sigma} (1 + \rho \sigma_{jkl}^2)^{-\xi} \tag{3.48}$$

Figure 3.7 shows the derivative of this penalty term with respect to the variance (σ_{jkl}^2). The gradient is negative and becomes large for very small variances. A negative gradient used in the parameter update (Eq. (3.30)) will result in the variance becoming

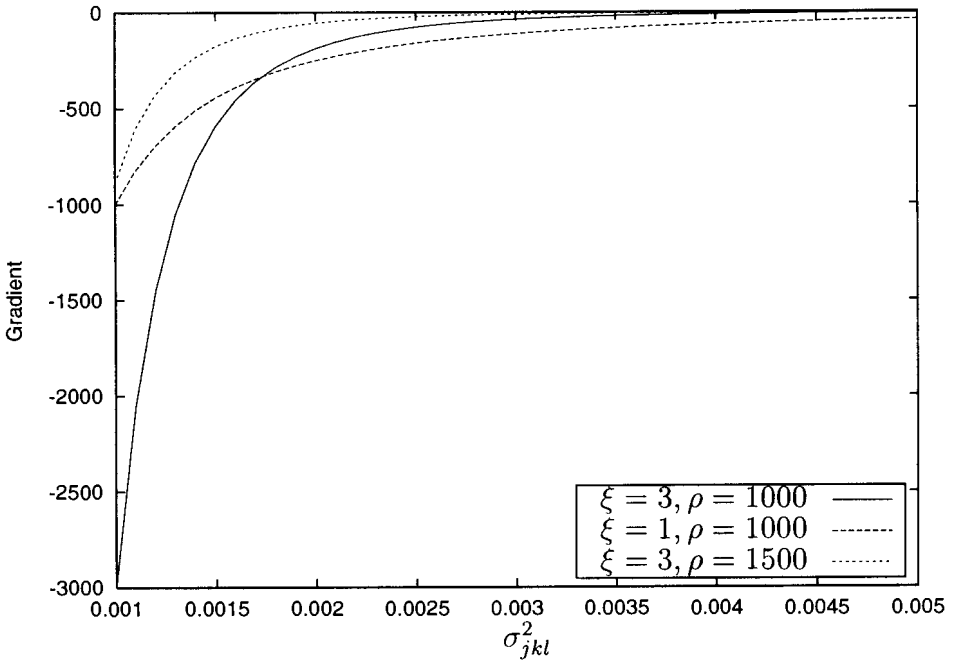
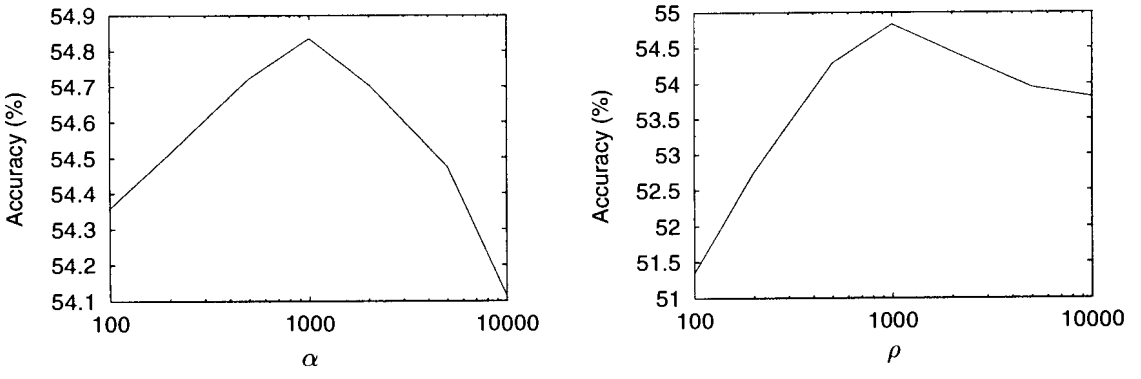


Figure 3.7: Gradient resulting from the variance penalty term.

larger. Note that if there were no other gradients resulting from the standard MCE update, the variances would continue to grow larger.

The values α, ρ and ξ are constants that are empirically determined. As seen in Figure 3.7, the constants ρ and ξ determine the form of the penalty term. The constant α is a scaling factor and is used to increase the contribution of the penalty term independently of the variance value. One has been added to $\rho\sigma^2_{jkl}$ to ensure that the gradient is finite for $\sigma_{jkl} = 0$.

This results in what could be called "precision decay", thereby ensuring that variances do not become too small. This has the indirect consequence of reducing overfitting. It was empirically found that $\xi = 3$ produced the best results. Figure 3.8 shows the performance of the MCE algorithm using the variance penalty term (MCE+VPEN) versus the parameters α and ρ when using the small training set T_S. Peak performance of 54.8% is attained when using $\alpha = 1000$ and $\rho = 1000$ (versus 53.8% without the penalty term).



(a) (b)

Figure 3.8: Testing set performance for the modified MCE with variance penalty (MCE+VPEN) using the small training set T_S, (a) versus α for $\rho = 1000$ and (b) versus ρ for $\alpha = 1000$

Figure 3.9 shows the advantage of using the above penalty term. Here, the parameter values $\xi = 3$, $\rho = 1000$ and $\alpha = 1000$ have been used, with no sigmoid loss function. The results are similar when using a sigmoid loss function. The test set results are better (2.1% relative improvement in error rate), while the training set performance has decreased.

Table 3.4 presents the results of using a sigmoid function versus not using it when using the variance penalty term with the MCE algorithm. Interestingly, a sigmoid function works better when the full training set is used, while the MCE algorithm without a sigmoid loss function performs better when the small training set is used. Note that the optimal values for parameters α and ρ differ for each configuration and were empirically determined for each case.

Table 3.4: Accuracy results for the MCE algorithm (sigmoid versus no sigmoid) when using the variance penalty term

Training set	sigmoid $\gamma = 0.01$	no sigmoid
T	65.4	64.7
T_S	53.8	54.8

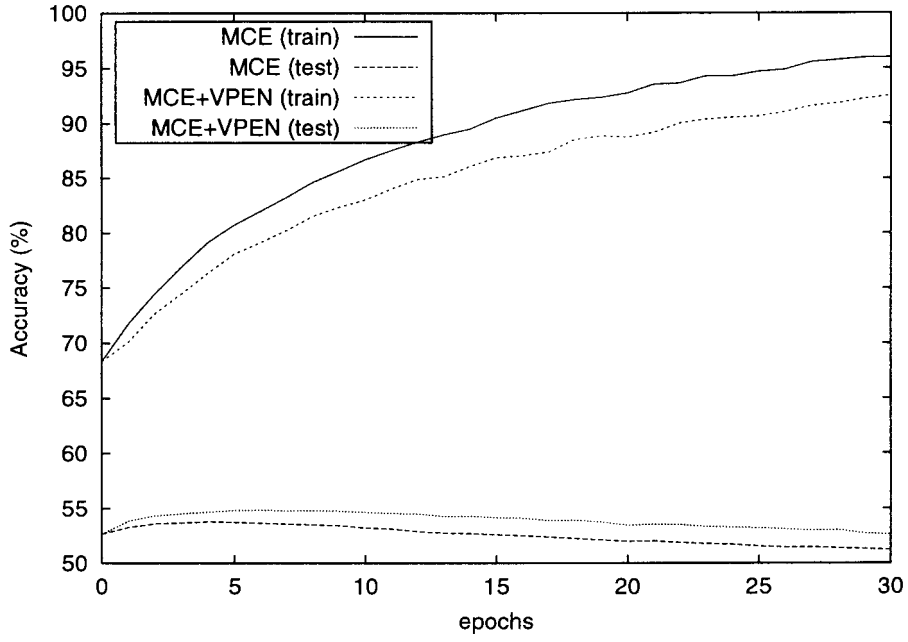


Figure 3.9: Training and testing set performance for standard string MCE and modified MCE with variance penalty (MCE+VPEN), no sigmoid loss

Weighted likelihood term

One potential problem with string-level MCE is that parts of the training string that do not result in errors are effectively ignored and therefore do not reinforce the associated parameters. Focusing solely on errors will result in overspecialization. Adding a weighted likelihood term (of the correct class) to the MCE loss function may therefore reduce overtraining. This will tend to reinforce correct substrings, while still penalizing errors. The misclassification measure in Eq. (3.13) then becomes

$$d(\mathbf{O}) = -(1+\kappa)\ln P_i(\mathbf{O};\theta_i) + \ln[\frac{1}{N}\sum_{j,j\neq i}^{N} e^{\eta \ln P_j(\mathbf{O};\theta_j)}]^{1/\eta}, \tag{3.49}$$

where κ is the weighting of the additional likelihood term, $P_i(\mathbf{O};\theta_i)$. This modification has been chosen so as to increase the gradient resulting from correct substrings by a factor κ. However, when using a sigmoid loss function, no modification (to misclassification measure or loss function) can be made that will increase the gradient associated

with correct substrings by a uniform factor κ, while not affecting that associated with incorrect substrings. Such a modification can therefore not be mathematically justified when using a smoothed zero-one loss function. It can, however, be implemented as a simple heuristic where the gradient for the correct class is simply multiplied by a weighting factor $(1 + \kappa)$.

Figure 3.10 presents the results for the MCE algorithm using the weighted likelihood term (MCE+WL) for the small training set (T_S). Significantly, using a sigmoid function does not work nearly as well as the algorithm without a sigmoid loss function when using the weighted likelihood term. Peak performance of 55.0% is attained when using a weight $\kappa = 1.5$, for the algorithm without a sigmoid loss function (versus 53.8% with $\kappa = 0$).

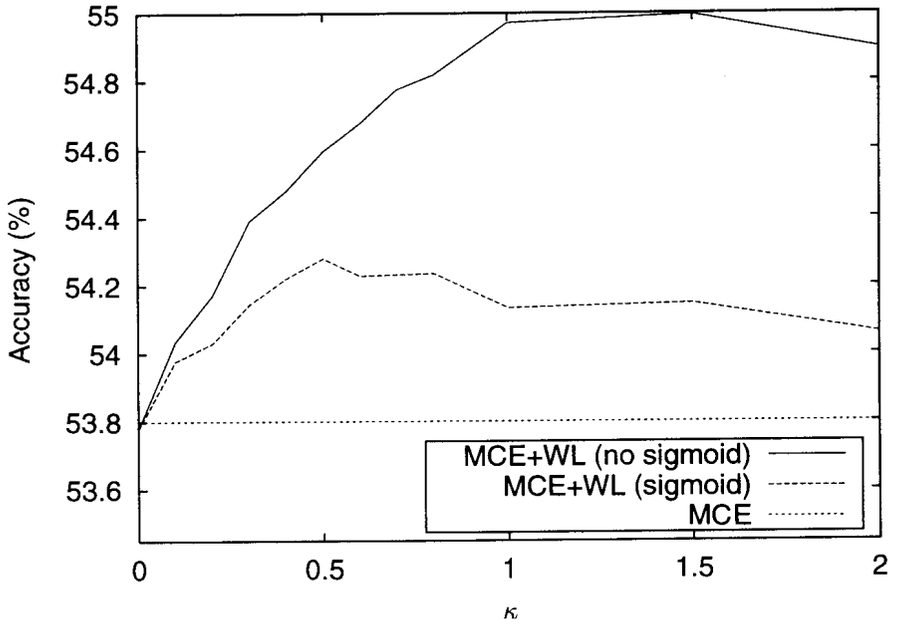


Figure 3.10: Testing set performance for MCE with the weighted likelihood term (MCE+WL) plotted versus κ.

Figure 3.11 shows the performance of the MCE algorithm with and without using the weighted likelihood term versus the number of training epochs, for the small training set. A sigmoid function is not used. Maximum testing set performance is better when using the weighted likelihood term, but more significantly, it stops degradation

of performance after peaking (preferred result (c) in Figure 3.6). Results when using a sigmoid loss function are similar with a slightly worse error rate being achieved.

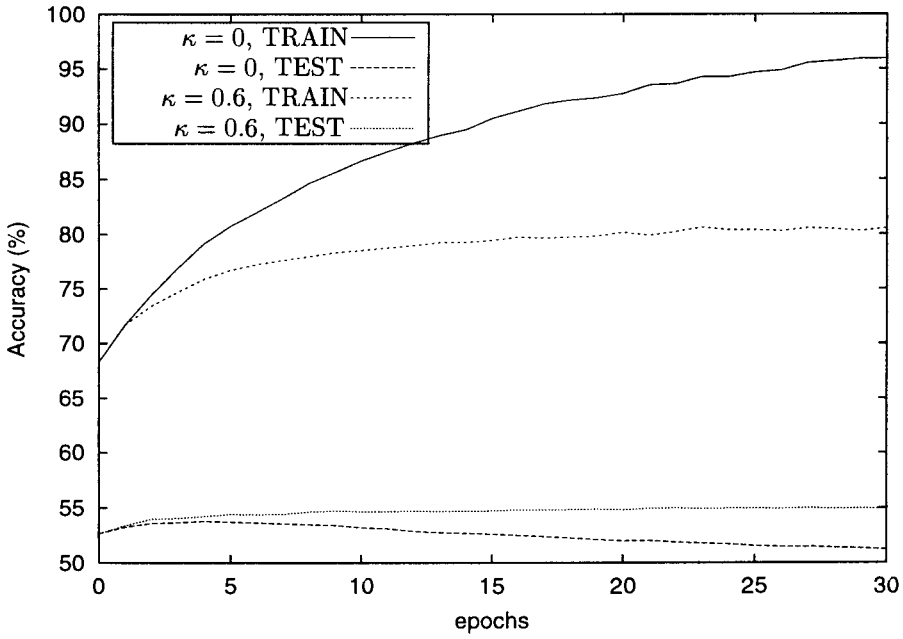


Figure 3.11: Training and testing set performance for standard string MCE and modified MCE with the weighted likelihood term (MCE+WL), sigmoid loss function not used

Table 3.5 presents the results when using the weighted likelihood term with the MCE algorithm for both the small and full training datasets. Here, in conjunction with the weighted likelihood term, using no sigmoid function performs best for both datasets. This is due to the fact that the implementation thereof for the algorithm using a sigmoid loss function is not mathematically justified and is merely a heuristic implementation thereof.

Table 3.5: Accuracy results for the MCE algorithms using the weighted likelihood term (MCE+WL)

Training set	sigmoid $\gamma = 0.01$	no sigmoid
T	64.7	65.1
T_S	54.3	55.0

Word MCE

Presenting arbitrarily long strings to the string-level MCE algorithm is not optimal. An error at a specific point in time will potentially result in the incorrect segmentation at that time (not just incorrect labelling), and such an error in segmentation will therefore influence the recognition and segmentation of subsequent acoustic units. The usage of a language model will also tend to result in an error at any given point resulting in further errors later in the utterance. Errors occurring earlier during recognition of a string therefore influence the recognition for the rest of the string. Our confidence in the accuracy of segmentation and classification after an error has occurred will therefore tend to be low. As the N-best string outputs from the recognizer are used as discriminative training examples, the number of incorrect strings are limited. Most of these "incorrect" strings differ only in a few places, resulting in only a few potential errors being addressed during discriminative training.

To improve the above, presenting smaller word-based strings to the string-level MCE algorithm is investigated. This is particularly appropriate when training speech recognizers on speech databases which have long sentences. This, however, requires that one has a dataset which is also labeled at word level.

A sentence would therefore be presented to the string-level MCE algorithm word by word in isolation. The N-best hypotheses (string of phones) would therefore be generated for each word individually (using the relevant part of the utterance) and used to determine the MCE gradients and updates. This is as opposed to the standard string-level MCE algorithm where the N-best hypotheses are generated for the entire sentence.

Figure 3.12 compares results for sentence- and word-based MCE for the small training set, plotted versus the number of training epochs. A sigmoid loss function is not used. The improvement in performance is marked, resulting in a 7.4% relative reduction in error rate. Another advantage here is that the usage of word-based string MCE limits

degradation of performance after a maximum is reached.

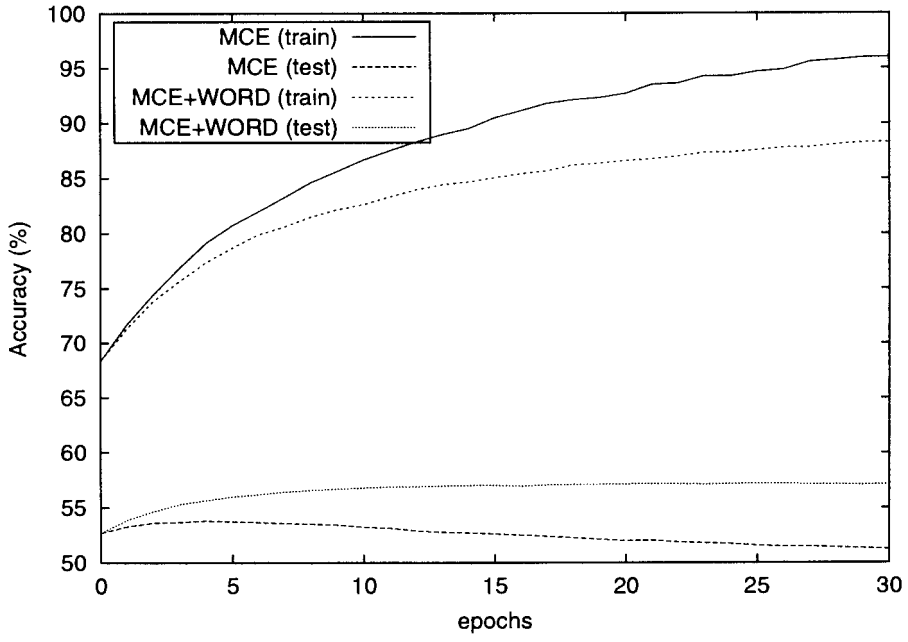


Figure 3.12: Training and testing set performance for standard string MCE and word-string MCE when using the small training set T_S (no sigmoid loss)

The results for both small and full training sets are presented in Table 3.6. Here, using a sigmoid loss function proves to be marginally better (and probably not significant) when the small training set (T_S) is used. This is, however, not the case when the full training set (T) is used, where using a sigmoid loss function produced a recognition accuracy which is marginally worse than that attained when not using a sigmoid loss function. Improvement in error rates relative to that obtained by standard MCE are significant (accuracy of 66.7% versus 64.7%), irrespective of whether a sigmoid loss function is used or not.

3.4.5 Summary and discussion of results

Table 3.7 gives a summary of the results when using the different modifications proposed for MCE in this chapter. The full training set (T) is used. MCE training alone produces a 17.7% *relative* reduction in error rate over baseline maximum likelihood

Table 3.6: Accuracy results for the word-based string MCE algorithms (MCE+WORD). Relative improvement in error rate compared to standard MCE are given in brackets

Training set	sigmoid $\gamma = 0.01$	no sigmoid
T	66.3 (5.1%)	66.7 (5.9%)
T_S	57.9 (8.9%)	57.2 (7.4%)

(ML) training. However, employing the modifications results in a relative reduction in error of up to 23.3% being attained over ML training.

Table 3.7: Summary of phoneme error rate results for MCE and modifications on the full training set T

Training method	Phoneme error rate %	
Baseline (ML)	43.0	
	sigmoid	no sigmoid
MCE	35.5	35.4
MCE+VPEN	34.6	35.3
MCE+WL	35.3	34.9
MCE+WORD	33.7	33.3
MCE+WORD+VPEN	33.3	**33.0**
MCE+WORD+WL	33.7	33.3

Table 3.8 gives a summary of the results when using the different modifications and the small training set (T_S) is used. Here, standard string-level MCE only results in a 2.5% relative reduction in error rate over baseline maximum likelihood (ML). However, employing the proposed modifications results in a relative reduction in error rate of up to 12.2% being attained. The modifications have more of an effect when less training data is available and overtraining is more prevalent.

Tables 3.7 and 3.8 provide results obtained when combining the word-based string MCE algorithm and the other modifications (penalty and weighted likelihood). Unfortu-

Table 3.8: Summary of phoneme error rate results for MCE and modifications on the small training set T_S

Training method	Phoneme error rate %	
Baseline (ML)	47.4	
	sigmoid	no sigmoid
MCE	46.2	46.2
MCE+VPEN	46.2	45.2
MCE+WL	45.7	45.0
MCE+WORD	42.1	42.8
MCE+WORD+VPEN	**41.6**	42.2
MCE+WORD+WL	42.1	42.8

nately, the weighted likelihood term fails to improve upon the performance of the word-based string-level MCE algorithm for either of the two datasets (MCE+WORD+WL versus MCE+WORD in the tables). This indicates that the effect of the two modifications is similar, which can be seen in the results presented for the individual procedures earlier. Both, for example, reduce degradation in performance after maximum testing set performance is reached. Limited improvements in performance were obtained when the variance penalty term was combined with the word-based string-level MCE algorithm.

Significant improvements in performance on the testing sets are obtained using the modifications to MCE as proposed. The modifications proposed are relatively simple to implement and limit overspecialization to some degree. The additional computational expense resulting from the use of the proposed modifications is very small and is not measurable. Although variation in performance did result from the use or non-use of a sigmoid function, there is little evidence to suggest that any one of the two possibilities is a better choice, with the resultant variation in performance generally being relatively small compared to the improvements in error rate due to the modifications.

3.5 Summary

This chapter described the MCE algorithm and its usage within a continuous speech recognition framework. The algorithm performs gradient descent on a criterion function which is a close approximation of the classification error. For the TIMIT database, usage of the MCE criterion resulted in significant gains in performance over the standard ML training procedure.

The effect of a smoothed zero-one loss function was discussed and experimentally determined for the TIMIT dataset; where it was found that small values of the sigmoid parameter γ performed best. Furthermore, the need for a zero-one loss function was questioned and the conclusion was reached that there is little evidence that there is an advantage or disadvantage to using a smoothed zero-one loss function. This result was used later in modifying the MCE criterion (adding a weighted likelihood term), where the modification could not be mathematically justified when a non-linear loss function was used.

Overtraining within the MCE framework was discussed and three modifications were proposed. The first modification attempted to stop the mixture variances from becoming very small, which results when little data is available. The second modification added a weighted likelihood term to the MCE criterion, thereby reinforcing correct substrings, as well as improving discrimination for incorrect substrings. Finally, a word-based string-level MCE algorithm was proposed, in which smaller word-based substrings were used, instead of the the entire string. Significant gains in performance resulted when using these modifications with the TIMIT database.

Chapter 4

Bayesian adaptation

In the previous chapter the usage of the MCE procedure in sparse data scenarios was investigated. In certain scenarios, however, one has a reasonable amount of non-task-specific training data available which can be used for training purposes. This chapter therefore investigates the usage of adaptation techniques that can use non-task-specific data as well as the limited task-specific data to create a better recognition system than could be attained though only using the task-specific data.

The standard *maximum a-posteriori* (MAP) adaptation technique is introduced and discussed. The MAP algorithm, however, makes assumptions about the form of the prior probability distribution used. This can be a problem when the prior is relatively complex. Two new adaptation algorithms are therefore proposed, a *gradient-based* MAP algorithm and an MCE-based adaptation algorithm. These adaptation algorithms make no assumptions about the form of the prior distribution used.

Section 4.1 introduces the basic Bayesian theory necessary for the algorithms in this chapter. The choice of prior distribution is discussed in Section 4.2. Section 4.3 reviews the maximum *a posteriori* estimation (MAP) algorithm as proposed by Gauvain and Lee [45]. An alternative *gradient-based* method of obtaining the MAP estimate

is described in Section 4.4. A Bayesian inspired *modification* to the MCE training procedure is then proposed in Section 4.5. Finally, the different methods discussed in this chapter are experimentally compared in Section 4.6.

4.1 Introduction

We can define a probabilistic model $P(\mathbf{X}|\theta)$ for any random process $\mathbf{X}$, in which a set of parameters θ determine its probability distribution. $P(\mathbf{X}|\theta)$ is called the *likelihood function*. We however want to infer θ in our probabilistic model, using data which we have obtained.

The result of Bayesian learning is a probability distribution $P(\theta|\mathbf{X})$ which expresses our beliefs of how likely individual parameters θ are, given the training data $\mathbf{X}$. $P(\theta|\mathbf{X})$ is called the *posterior distribution*. When classifying an unknown observation, the probability that the unknown observation was generated by the same process as that which generated the observations for a given class must be calculated. This is done by integrating the likelihood function with respect to the posterior distribution. Section 4.1.3 will present a more detailed discussion of this process.

Bayesian methods can be used for the inference of parameter values in a model, given the data. We can, however, also use Bayesian methods for the purpose of model comparison, where preferences are assigned to alternative models of differing complexity. David Mackay [72] focused primarily on the usage of Bayesian methods for the training and comparison of neural network models. Mackay used Bayesian methods to compare models of differing complexity and topology. Most people would include the above two uses of Bayesian methods in the data modeling process.

A further application of Bayesian methods is in the adaptation of existing models. An example of this in speech recognition is maximum a-posteriori (MAP) parameter estimation. MAP is a point estimate and one does not integrate with respect to the

posterior distribution. Many would therefore claim that MAP is not a true Bayesian method. If, however, one assumes that the posterior distribution is sufficiently peaked about the most probable θ (maximum *a-posteriori* probability), then the MAP procedure is a reasonable approximation of Bayesian learning. MAP can be used for several purposes, including parameter smoothing and adaptation. Parameter smoothing applies extra constraints to the model parameters so as to reduce the effect of insufficient training data. This can be achieved using MAP by incorporating vague heuristic information in the prior distribution. In adaptation, however, non-task-specific information is available and is used to determine the prior distribution used in MAP estimation.

The remainder of this section will summarize the pertinent Bayesian theory used in this chapter. For a more complete introductory text on Bayesian statistics, the reader is referred to Box and Tiao [13] and DeGroot [27]. The theory and discussions in this section will be biased towards speech recognition applications of Bayesian adaptation.

4.1.1 Bayes' theorem

Given a vector $\mathbf{y} = (y_1, \ldots, y_n)$ of n observations, with probability distribution $P(\mathbf{y}|\theta)$, which depends on the k parameters $\theta^T = (\theta_1, \ldots, \theta_k)$ with probability distribution $P(\theta)$, then given the observed data $\mathbf{y}$, the conditional distribution of θ is

$$P(\theta|\mathbf{y}) = \frac{P(\mathbf{y}|\theta)P(\theta)}{P(\mathbf{y})}. \tag{4.1}$$

The denominator in Eq. (4.1), $P(\mathbf{y})$, is a normalizing factor, which ensures that the integral of $P(\theta|\mathbf{y})$ is equal to one. It can be written as follows:

$$P(\mathbf{y}) = \int P(\mathbf{y}|\theta)P(\theta)d\theta. \tag{4.2}$$

Equation (4.1) is referred to as Bayes' theorem. The distribution $P(\theta)$, is called the

prior distribution and expresses what is known about the model parameters before any data is observed. The *posterior* distribution $P(\theta|\mathbf{y})$, tells us what is known about the model parameters, given that data has been observed. In what follows, the prior distribution and posterior distribution will sometimes simply be referred to as the "prior" and "posterior" respectively.

The distribution $P(\mathbf{y}|\theta)$ is often referred to as the data *likelihood* and can be written $L(\theta|\mathbf{y})$. Then in effect, $P(\mathbf{y}|\theta)$ is regarded as a function of $\mathbf{y}$ and not of θ.

In many Bayesian methods, the normalizing constant is not necessary and Eq. (4.1) is written as

$$P(\theta|\mathbf{y}) \propto L(\theta|\mathbf{y})P(\theta). \tag{4.3}$$

4.1.2 Sequential nature of Bayes' theorem

If we assume that the observations are independent, then we can write Bayes' theorem as follows

$$P(\theta|\mathbf{y}) \propto P(\theta)\prod_{i=1}^{n} L(\theta|\mathbf{y}^{(i)}) = \left[P(\theta)\prod_{i=1}^{k} L(\theta|\mathbf{y}^{(i)})\right]\prod_{i=k+1}^{n} L(\theta|\mathbf{y}^{(i)}) \tag{4.4}$$

$$\propto P(\theta|\mathbf{y}_1,\ldots,\mathbf{y}_k)\prod_{i=k+1}^{n} L(\theta|\mathbf{y}^{(i)}) \qquad (k<n). \tag{4.5}$$

Equation (4.4) is exactly the same as Eq. (4.5), except that $P(\theta|\mathbf{y}_1,\ldots,\mathbf{y}_k)$, the posterior distribution of θ given $\mathbf{y}_1,\ldots,\mathbf{y}_k$, now acts as the prior distribution for $\mathbf{y}_{k+1},\ldots,\mathbf{y}_n$.

Bayes' theorem, therefore describes the process of learning as data becomes available. We can therefore, as Eq. (4.5) suggests, compute the posterior for a given set of data

and then use that posterior as a "prior" when more data becomes available. This result is of utmost importance to the methods proposed later in this chapter.

4.1.3 Bayesian learning and prediction

The result of Bayesian learning is a probability distribution (posterior) which expresses our beliefs of how likely individual parameters values are. This is crucial, as it allows learning to be performed using probability theory.

In a Bayesian approach to HMM parameter estimation and recognition, the objective is to find a predictive distribution ($P(\mathbf{x}|\mathbf{X}) = \int P(\mathbf{x}|\theta)P(\theta|\mathbf{X})d\theta$) for an unknown utterance, given the utterance observations, as well as the training observations. Let the observation sequence for the ith example be written as $\mathbf{O}_i$. For n training examples $\mathbf{O} = (\mathbf{O}_1, \ldots, \mathbf{O}_n)$, Bayes' theorem (Eq. (4.1)) can be written as

$$\begin{aligned} P(\theta|\mathbf{O}) &= \frac{P(\mathbf{O}|\theta)P(\theta)}{P(\mathbf{O})} \\ &\propto P(\mathbf{O}|\theta)P(\theta). \end{aligned} \tag{4.6}$$

Assuming independence of the observations we can write the likelihood as follows:

$$P(\mathbf{O}_i|\theta) = \prod_{j=1}^{n_u} P(\mathbf{o}_{ij}|\theta), \tag{4.7}$$

where n_u is the number of observations in the training utterance $\mathbf{O}_i$. In a Bayesian framework, when we wish to classify an unknown input, we need to calculate the following probability,

$$P(\mathbf{O}_{unknown}|\mathbf{O}_1^{(i)}, \ldots, \mathbf{O}_n^{(i)}) = \int P(\mathbf{O}_{unknown}|\theta)P(\theta|\mathbf{O}_1^{(i)}, \ldots, \mathbf{O}_n^{(i)})d\theta, \tag{4.8}$$

where i is the class and $\mathbf{O}_{unknown}$ is the unknown observation sequence. The classifier decision is the class resulting in the highest value of Eq. (4.8), i.e.

$$C(\mathbf{O}_{unknown}) = i \quad \text{where} \quad i = \underset{j}{argmax} P(\mathbf{O}_{unknown}|\mathbf{O}_1^{(j)}, \ldots, \mathbf{O}_n^{(j)}), \tag{4.9}$$

where $C(\mathbf{O}_{unknown})$ is the classifier's decision for the unknown observation.

4.1.4 Maximum *a-posteriori* probability estimate

Assuming the posterior is sufficiently peaked around the most probable point (θ_{MAP}), we can approximate Eq. (4.8) as

$$P(\mathbf{O}_{unknown}|\mathbf{O}_1, \ldots, \mathbf{O}_n) \approx P(\mathbf{O}_{unknown}|\theta_{MAP}), \tag{4.10}$$

where θ_{MAP} is the Maximum *a-posteriori* (MAP) estimate of the parameter vector θ.

The MAP point is the set of parameters that maximize Eq. (4.6), or

$$\begin{aligned} \theta_{MAP} &= \underset{\theta}{argmax} P(\theta|\mathbf{O}_1, \ldots, \mathbf{O}_n) \\ &= \underset{\theta}{argmax} \Big[P(\mathbf{O}_1, \ldots, \mathbf{O}_n|\theta) P(\theta) \Big]. \end{aligned} \tag{4.11}$$

If we had no prior knowledge about θ, then we would choose a non-informative (improper) prior to be used in Eq. (4.11), i.e., $P(\theta) = constant$. Equation (4.11) then reduces to the normal maximum likelihood (ML) formulation.

4.1.5 MAP adaptation in speech recognition

This section presents a concise literature survey of the usage of the MAP procedure for HMM parameter estimation in speech recognition.

Lee *et al.* [67] introduced a MAP algorithm, where the parameters of multivariate Gaussian state observation densities of HMM models were adapted for speaker adaptation. They showed that for an alpha digit task, with only a small amount of speaker specific data, their MAP estimated HMM gave better results than an ML estimated HMM.

Gauvain and Lee [43] extended the MAP formulation for HMMs to handle parameters of *mixtures* of Gaussian densities. It was shown that MAP estimation can be used for parameter smoothing, speaker adaptation, speaker clustering and corrective training. Gauvain and Lee [45] later presented a theoretical framework for MAP estimation for HMMs with Gaussian mixture state densities, where they proposed using an expectation-maximization (EM) approach to finding the MAP estimate. In this work, the MAP formulation was also extended to include the estimation of the transition probabilities and initial state probabilities. The application of their MAP algorithm to corrective training and parameter smoothing [44] and speaker adaptation [66] were also reported.

Huo *et al.* [53, 52] studied the usage of MAP estimation for semi-continuous (or tied mixture) HMMs. Zavaliagkos *et al.* [120] also investigated using various degrees of parameter tying, so as to force MAP to adapt parameters for which adaptation data is not available.

The MAP estimation algorithm can be used for various purposes, including parameter smoothing [44], speaker adaptation [67, 66, 32], dialect adaptation [41] and cross-language adaptation [85, 84]. In this chapter, we are primarily interested in the usage of the MAP estimation algorithm for adaptation purposes as opposed to applications

such as parameter smoothing and corrective training. Model adaptation is a process for adjusting seed models to create more specialized models using a small amount of adaptation data. Figure 4.1 illustrates an abstraction of the MAP adaptation algorithm as it would typically be used in speaker adaptation. The process is very similar for other adaptation applications of the algorithm, such as for cross-language adaptation.

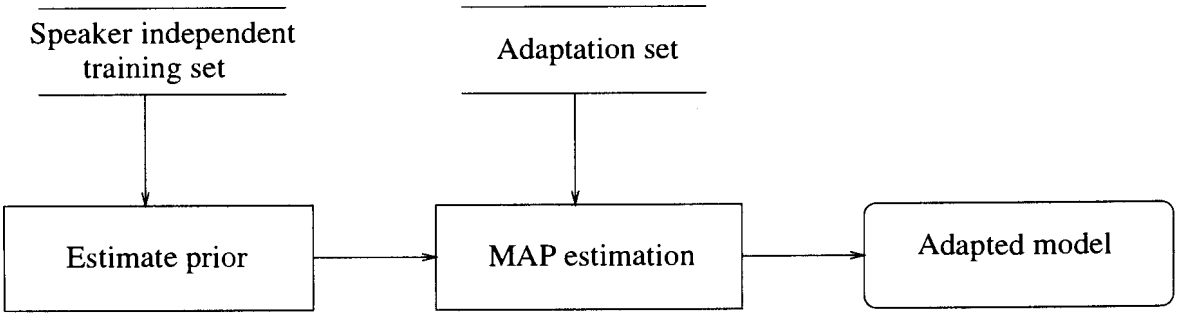


Figure 4.1: Abstraction of the typical MAP speaker adaptation process

4.2 Prior distributions

In this section, the choice of the prior density family is discussed. The prior distribution is an important part of any Bayesian method, as it expresses our knowledge about the distributions prior to any data being observed. It is especially important when there is little data available.

Prior distributions must therefore be chosen for the following HMM parameters:

- transition probabilities a_{ij} (Eq. (2.3)),
- mixture weights C_{jk} (Eq. (2.5)),
- Gaussian means μ_{jk} (Eq. (2.5)), and
- Gaussian variances Σ_{jk} (Eq. (2.5)).

Conjugate prior (see Appendix A) distributions have been chosen in this work, not only because of the convenient relationship between prior and posterior, but also due to their usage in the literature. If the prior distribution of θ belongs to a conjugate family of distributions, then for a likelihood of a specific form, the posterior distribution of θ must also belong to the same family as that of the prior distribution. Note that with most of the approaches developed in this and later chapters it is not necessary nor even convenient to have conjugate prior distributions. It is, however, essential to use a conjugate prior for the application of the EM algorithm to the MAP estimation problem (Section 4.3).

The prior distribution for all the parameters of the Gaussian mixtures of the HMMs is chosen to be a normal-Wishart distribution, where the conditional of μ (mean) and R (precision matrix, with $R = \Sigma^{-1}$) are given as follows: The conditional distribution of μ (when $R = r$), is a multivariate normal distribution with mean vector m and precision matrix νr for $m \in \Re^D$ and $\nu > 0$. The marginal distribution of R is a Wishart distribution with α degrees of freedom and precision matrix τ. The priors for the transition probabilities and Gaussian mixture weights are chosen to be Dirichlet distributions. Table 4.1 summarizes the HMM parameters and their prior distributions.

Table 4.1: Summary of the HMM parameters and chosen prior distributions. A Wishart distribution is represented with $\mathcal{W}$ and a Dirichlet with $\mathcal{D}$

Parameter	Prior distribution	Prior parameters
$\mu_{jk}\|R_{jk} = r_{jk}$	$\mathcal{N}(m_{jk}, (\nu_{jk} r_{jk})^{-1/2})$	$mean:\ m_{jk},\quad precision:\ \nu_{jk} r_{jk}$
$R_{jk}\ (\Sigma_{jk}^{-1})$	$\mathcal{W}(n_{jk}, \tau_{jk})$	$degrees\ of\ freedom:\ n_{jk},\quad precision:\ \tau_{jk}$
c_j	$\mathcal{D}(\delta_j)$	δ_j
a_i	$\mathcal{D}(\alpha_i)$	α_i

The prior of the parameters (μ_{jk} and r_{jk}) of the Gaussian mixture component k of state j can therefore be written (from Eqs. (A.5) and (A.1)) as [27]:

$$g_{gaussian}(r_{jk}, \mu_{jk}|n_{jk}, \nu_{jk}, m_{jk}, \tau_{jk}) = (2\pi)^{-\frac{D}{2}}|\nu_{jk}r_{jk}|^{\frac{1}{2}}e^{-\frac{1}{2}\nu_{jk}(\mu_{jk}-m_{jk})^T r_{jk}(\mu_{jk}-m_{jk})} \\ c|\tau_{jk}|^{\frac{n_{jk}}{2}}|r_{jk}|^{\frac{(n-D-1)}{2}}e^{-\frac{1}{2}tr(r_{jk}\tau_{jk})}, \tag{4.12}$$

where $(n_{jk}, \nu_{jk}, m_{jk}, r_{jk})$ are the prior distribution parameters. The value c is a normalizing constant which ensures that the integral of the prior is equal to one. Assuming a diagonal covariance (or precision) matrix, the log of the Gaussian distribution prior (Eq. (4.12)) can be written as

$$\begin{aligned} log\ g_{gaussian}(\Sigma_{jk}, \mu_{jk}|n_{jk}, \nu_{jk}, m_{jk}, \tau_{jk}) = &-\frac{D}{2}log(2\pi\nu_{jk}) + log(c) \\ &- \frac{n_{jk}-D}{2}log\left(\prod_{l=1}^{D}\sigma_{jkl}^2\right) - \frac{1}{2}\nu\sum_{l=1}^{D}\sigma_{jkl}^{-2}(\mu_{jkl}-m_{jkl})^2 \\ &+ \frac{n_{jk}}{2}\cdot log\left(\prod_{l=1}^{D}\tau_{jkl}\right) - \frac{1}{2}\sum_{l=1}^{D}\frac{\tau_{jkl}}{\sigma_{jkl}^2}. \end{aligned} \tag{4.13}$$

The prior for the mixture weights can be written as:

$$g_{weight}(c_j|\delta_j) = \frac{\Gamma(\delta_{j1}+\delta_{j2}+\cdots+\delta_{jM})}{\Gamma(\delta_{j1})\Gamma(\delta_{j2})\cdots\Gamma(\delta_{jM})}c_{j1}^{\delta_{j1}-1}c_{j2}^{\delta_{j2}-1}\cdots c_{jM}^{\delta_{jM}-1}, \tag{4.14}$$

where δ_j is the parameter vector associated with the mixture weight priors for state j.

The prior for the transition probabilities can be written as:

$$g_{trans}(a_i|\alpha_i) = \frac{\Gamma(\alpha_{i1}+\alpha_{i2}+\cdots+\alpha_{iN})}{\Gamma(\alpha_{i1})\Gamma(\alpha_{i2})\cdots\Gamma(\alpha_{iN})}a_{i1}^{\alpha_{i1}-1}a_{i2}^{\alpha_{i2}-1}\cdots a_{ik}^{\alpha_{iN}-1}, \tag{4.15}$$

where α_i is the prior parameter vector associated with the transition probabilities a_i from state i and N being the number of states in the HMM.

If we assume independence for parameters of the Gaussians, mixture weights and transition probabilities, the joint prior density $g(\theta)$ for all the parameters of the HMM can be written as the product of the prior p.d.f's defined in Eqs. (4.12), (4.14) and (4.15), i.e.

$$g(\theta) = \prod_{i=1}^{N} \left(g_{trans}(a_i|\alpha_i) g_{weight}(c_i|\delta_i) \prod_{k=1}^{M} g_{gaussian}(\Sigma_{ik}, \mu_{ik}|n_{ik}, \nu_{ik}, m_{ik}, r_{ik}) \right), \quad (4.16)$$

where M is the number of Gaussian mixture components per state. The joint prior density for all HMMs can, assuming independence, be written as the product of the joint prior distributions (Eq. (4.16)) of the individual HMMs.

4.3 Expectation-Maximization MAP

Gauvain and Lee [45] introduced a method of estimating the MAP point for all the HMM parameters as defined in Table 4.1 using an expectation maximization (EM) approach. Their method and related theory will be briefly reviewed in this section. This algorithm will, for the rest of this thesis, often be referred to as simply the MAP algorithm.

MAP estimation is relatively simple if the family of p.d.f.'s $P(\cdot|\theta), \theta \in \Theta$ possesses a sufficient statistic of fixed dimension for the parameter θ we wish to estimate. A sufficient statistic T only exists if $P(\mathbf{O}|\theta)$ can be factored [27] as follows for all values of $\mathbf{O}$ and θ:

$$P(\mathbf{O}|\theta) = u(\mathbf{O})v(T(\mathbf{O}), \theta). \quad (4.17)$$

Here, the function u is positive and does not depend on θ, while the function v is non-negative and depends on the data (observations) $\mathbf{O}$ only through $T(\mathbf{O})$. For hidden

Markov models, due to the underlying hidden process, a sufficient statistic of fixed dimension does not exist. This is known as an "incomplete data" problem. When using hidden Markov models, the true state sequence $\mathbf{q} = \{q_1, \ldots, q_T\}$ and the true sequence of associated mixture components $\mathbf{l} = \{l_1, \ldots, l_t\}$ are not observed or known. As a result, the observed data $\mathbf{O}$ is not sufficient to be able to directly estimate the HMM parameters (*incomplete data*). For HMMs, the *complete data* $\mathbf{x}$ is the combination of the observations $\mathbf{O}$, state sequence $\mathbf{q}$ and mixture component sequence $\mathbf{l}$, i.e. $\mathbf{x} = (\mathbf{O}, \mathbf{q}, \mathbf{l})$.

The term *incomplete data* ($\mathbf{y}$) implies two sample spaces $\mathcal{X}$ and $\mathcal{Y}$, and a many-to-one mapping from $\mathcal{X}$ to $\mathcal{Y}$. Given $\mathbf{x} \in \mathcal{X}$ and $\mathbf{y} \in \mathcal{Y}$, where $\mathbf{x}$ is not observed directly but indirectly through the observed data $\mathbf{y}$, a mapping $\mathbf{x} \rightarrow \mathbf{y}(\mathbf{x})$ exists from $\mathcal{X}$ to $\mathcal{Y}$, with $\mathbf{x}$ known only to be in $\mathcal{X}(\mathbf{x})$, the subset of $\mathcal{X}$ determined by the equation $\mathbf{y} = \mathbf{y}(\mathbf{x})$. We refer to $\mathbf{x}$ as the "complete data". Note that there are many *complete-data* specifications $f(\mathbf{x}|\theta)$ that will generate a given *incomplete-data* specification $g(\mathbf{y}|\theta)$.

The expectation-maximization(EM) algorithm [5, 28, 71] discussed next is an iterative procedure for approximating ML estimates in cases involving *incomplete data.*

4.3.1 Expectation maximization

Expectation maximization, as described in this section, is typically used for maximum likelihood (ML) estimation. Expectation maximization re-estimation is based on an auxiliary function $Q(\theta, \bar{\theta})$ defined in terms of the current parameter set θ and the new parameter set $\bar{\theta}$. The auxiliary function is defined as the expectation of the *complete-data* log-likelihood $log[f(\mathbf{x}|\bar{\theta})]$ given the observed incomplete data $\mathbf{y}$ and the current parameter set θ, i.e.

$$Q(\theta, \bar{\theta}) = E[log\ f((\mathbf{x}|\bar{\theta})|\mathbf{y}, \theta)] \tag{4.18}$$

which exists for all pairs $(\theta, \bar{\theta})$. The EM iteration is defined as follows:

Expectation (E-step): Compute $Q(\theta, \bar{\theta})$.

Maximization (M-step): Choose $\bar{\theta}$ which maximizes $Q(\theta, \bar{\theta})$. We simply differentiate $Q(\theta, \bar{\theta})$ with respect to $\bar{\theta}$ and find a maximum (i.e. solve $\frac{\partial Q(\theta,\bar{\theta})}{\partial \bar{\theta}} = 0$).

For an HMM, the *complete-data* likelihood is the joint likelihood of $\mathbf{O}$, $\mathbf{q} = \{q_1, \ldots, q_T\}$ the unobserved state sequence and $\mathbf{l} = \{l_1, \ldots, l_t\}$ the unobserved sequence of associated mixture components ($\mathbf{y} = (\mathbf{O}, \mathbf{q}, \mathbf{l})$).

For an HMM, the auxiliary function $Q(\theta, \bar{\theta})$ in Eq. (4.18) takes the following form [71]:

$$Q(\theta, \bar{\theta}) = \sum_{\mathbf{q}} E[f(\mathbf{O}, \mathbf{q}|\theta) log f(\mathbf{O}, \mathbf{q}|\bar{\theta})] \tag{4.19}$$

where $f(\mathbf{O}, \mathbf{q}|\theta)$ is the probability of observing the data $\mathbf{O}$ for state sequence $\mathbf{q}$ given the parameter set θ. The utility of $Q(\theta, \bar{\theta})$ stems in part from the fact that if $Q(\theta, \bar{\theta}) > Q(\theta, \theta)$, then $f(\mathbf{O}|\bar{\theta}) > f(\mathbf{O}|\theta)$. This property is shown in Eqs. (4.20) and (4.21).

$$Q(\theta, \bar{\theta}) - Q(\theta, \theta) = \sum_{\mathbf{q}} E\left[f(\mathbf{O}, \mathbf{q}|\theta) log\left(\frac{f(\mathbf{O}, \mathbf{q}|\bar{\theta})}{f(\mathbf{O}, \mathbf{q}|\theta)}\right)\right] \tag{4.20}$$

Using the inequality $log(x) \leq x - 1$, we can write Eq. (4.20) as:

$$\begin{aligned} Q(\theta, \bar{\theta}) - Q(\theta, \theta) &\leq \sum_{\mathbf{q}} E\left[f(\mathbf{O}, \mathbf{q}|\theta)\left(\frac{f(\mathbf{O}, \mathbf{q}|\bar{\theta})}{f(\mathbf{O}, \mathbf{q}|\theta)} - 1\right)\right] \\ &\leq f(\mathbf{O}|\bar{\theta}) - f(\mathbf{O}|\theta). \end{aligned} \tag{4.21}$$

Hence, $Q(\theta, \bar{\theta}) > Q(\theta, \theta)$ implies that $f(\mathbf{O}|\bar{\theta}) > f(\mathbf{O}|\theta)$. Therefore, when $Q(\theta, \bar{\theta})$ as defined in Eq. (4.19) is maximized in the maximization step, then $f(\mathbf{O}|\bar{\theta})$ will also

be maximized. The EM algorithm described here will be used in the next section to iteratively obtain the MAP estimate.

4.3.2 MAP estimate for HMMs

Although the expectation maximization procedure (Eq. (4.19)) is typically used to find the ML estimate, we wish to use it to obtain the MAP estimate. Dempster *et al.* [28] pointed out that the posterior mode (MAP point) can also be estimated using the EM algorithm by maximizing $Q(\theta, \bar{\theta}) + log[g(\theta)]$ at the M-step of each EM iteration. It was also shown that $log[f(\mathbf{x}|\theta)] + log[g(\theta)]$ increases at each EM iteration and an expression for the rate of convergence was provided.

It is relatively straightforward to show [45] that the auxiliary function of the EM algorithm applied to the MAP problem can be decomposed into the sum of three auxiliary functions, $Q_\pi(\bar{\pi}, \theta)$, $Q_A(\bar{A}, \theta)$ and $Q_\Upsilon(\Upsilon, \theta)$, with $\Upsilon = (\bar{c}, \bar{\mu}, \bar{\Sigma})$. The three auxiliary functions are maximized independently, resulting in the following re-estimation formulas:

$$\bar{\pi}_i = \frac{(\alpha_{0i} - 1) + \gamma_0(i)}{\sum_{j=1}^{N}(\alpha_{0i} - 1) + \sum_{j=1}^{N}\gamma_0(j)} \tag{4.22}$$

$$\bar{a}_{ij} = \frac{(\alpha_{ij} - 1) + \sum_{t=1}^{T}\xi_t(i,j)}{\sum_{j=1}^{N}(\alpha_{ij} - 1) + \sum_{j=1}^{N}\sum_{t=1}^{T}\xi_t(i,j)} \tag{4.23}$$

$$\bar{c}_{ij} = \frac{(\delta_{ik} - 1) + \sum_{t=1}^{T} w_{ikt}}{\sum_{k=1}^{M}(\delta_{ik} - 1) + \sum_{k=1}^{M}\sum_{t=1}^{T} w_{ikt}} \tag{4.24}$$

$$\bar{\mu}_{ik} = \frac{\nu_{ik} m_{ik} + \sum_{t=1}^{T} w_{ikt}\mathbf{o}_t}{\nu_{ik} + \sum_{t=1}^{T} w_{ikt}} \tag{4.25}$$

$$\bar{\Sigma}_{ik} = \frac{\tau_{ik} + \sum_{t-1}^{T} w_{ikt}(\mathbf{o}_t - \mu_{ik})(\mathbf{o}_t - \mu_{ik})^T + \nu_{ik}(m_{ik} - \mu_{ik})(m_{ik} - \mu_{ik})^T}{(n_{ik} - D) + \sum_{t=1}^{T} w_{ikt}}, \tag{4.26}$$

where w_{ikt} is defined as

$$w_{ikt} = \gamma_t(i) \frac{c_{ik}\mathcal{N}(\mathbf{o}_t|\mu_{ik}, \Sigma_{ik})}{\sum_{l=1}^{M} c_{ik}\mathcal{N}(\mathbf{o}_t|\mu_{ik}, \Sigma_{ik})}. \tag{4.27}$$

The values $\gamma_t(i)$, $\gamma_t(i,j)$ and $\xi_t(i,j)$ are obtained using the forward-backward algorithm [90, p. 334].

Segmental MAP estimate

Instead of maximizing $f(\theta|\mathbf{O})$ (using the forward-backward algorithm), we can, using Viterbi alignment, maximize $f(\theta, \mathbf{q}|\mathbf{O})$, the joint posterior density of the parameter set θ and the state sequence $\mathbf{q}$. It can easily be shown, as for the segmental k-means algorithm [61], that alternate maximization over $\mathbf{q}$ and θ results in a non-decreasing $f(\theta^{(n)}, \mathbf{q}^{(n)}|\mathbf{O})$, with

$$\mathbf{q}^{(n+1)} = \underset{\mathbf{q}}{argmax} f(\mathbf{O}, \mathbf{q}|\theta^{(n)}) \tag{4.28}$$

$$\theta^{(n+1)} = \underset{\theta}{argmax} f(\mathbf{O}, \mathbf{q}^{(n+1)}|\theta). \tag{4.29}$$

The most likely state sequence in Eq. (4.28) is obtained using the Viterbi algorithm. The EM algorithm can once again be used to perform the maximization in Eq. (4.29). The re-estimation formulae (4.22) to (4.26) used in the forward-backward MAP estimate are used here, however, the probabilities $\xi_t(i,j)$, $\gamma_t(i)$ and $\gamma_t(j,k)$ are obtained from the best state sequence (and not using the forward-backward algorithm) as follows:

$$\xi_t(i,j) = \delta(\mathbf{q}_t - i)\delta(\mathbf{q}_{t+1} - j), \tag{4.30}$$

$$\gamma_t(i) = \delta(\mathbf{q}_t - i), \tag{4.31}$$

$$\text{and } \gamma_t(j,k) = \delta(\mathbf{q}_t - j)\frac{c_{jk}\mathcal{N}(\mathbf{o}_t, \mu_{jk}, \Sigma_{jk})}{b_j(\mathbf{o}_t)}, \tag{4.32}$$

with δ being the Kronecker delta function.

The segmental MAP approach described above has been implemented and used in this work. An embedded version of the MAP algorithm has also been implemented and used, which uses the trellis search algorithm (Section 2.1.5) to obtain the best state and HMM sequence for a given utterance. The embedded MAP algorithm uses a search to automatically segment and label the acoustic units, and therefore does not rely on a labelled database being available. This is essential for databases, such as TIDIGITS, which are not manually labelled either at phoneme or word levels.

4.3.3 Prior density estimation

In Section 4.2 the form of the prior distributions and the distribution parameters were described. In this section, the estimation of the prior parameters will be discussed. The prior distribution should, in a true Bayesian approach, incorporate *a-priori* knowledge of the parameters we are attempting to estimate.

Two methods of estimating the prior parameters will be discussed here, namely empirical Bayes and a simpler method typically used in MAP adaptation. The empirical Bayes method is not used, but is included so that the two methods can be compared. When estimating the prior parameters we will only be using the prior data. The adaptation data cannot be used in any way when determining the prior parameters.

Empirical Bayes methods

In Empirical Bayesian methods [73, 97] the prior distribution $g(\theta)$ is estimated by finding a distribution function g that satisfies the relationship

$$h(\mathbf{O}) = \int f(\mathbf{O}|\theta)g(\theta)d\theta, \tag{4.33}$$

where $h(\mathbf{O})$ is the probability distribution of the data ($P(\mathbf{y})$ in Eq. (4.2)).

It is, however, necessary to ensure that the distribution $g(\theta)$ obtained is unique. The search for $g(\theta)$ could be an almost impossible task if we don't choose $g(\theta)$ to be part of a given parametric family $g(\theta; \alpha, \beta, \ldots)$ of distributions, where $\alpha, \beta, \ldots$ are the unknown prior parameters.

In this case, we do not know $h(\mathbf{O})$ exactly, but estimate an empirical distribution function $h_n(\mathbf{O})$ obtained from a sample of n observations on the random variable whose distribution function is $h(\mathbf{O})$. We therefore have the approximate relationship

$$h_n(\mathbf{O}) \approx \int f(\mathbf{O}|\theta)g(\theta)d\theta. \tag{4.34}$$

There are several methods for solving Eq. (4.34), i.e. finding $g(\theta)$, including maximum-likelihood (ML) and the method of moments. The solution of Eq. (4.34) is not a trivial task.

Though the empirical Bayes approach is not implemented or used for the MAP estimation in this work, it is important to understand the empirical Bayes approach and we therefore digress at this point to study a simple example taken from [73].

Given observed data $\mathbf{x}$, a model $f(\mathbf{O}|\theta) = f(\mathbf{x}|M)$, being a simple Normal distribution with mean M and standard deviation of 1 and a prior $g(M)$ that is a Normal

distribution with mean μ and variance σ^2, i.e.

$$f(\mathbf{x}|M) = \mathcal{N}(M, 1),$$
$$g(M) = \mathcal{N}(\mu, \sigma^2),$$

then $h_n(\mathbf{O}) = f_G(\mathbf{x}) = \int f(x|M)g(M;\mu,\sigma)dM = \mathcal{N}(\mu, 1+\sigma^2)$. The maximum likelihood estimates of the prior parameters $\tilde{\mu}$ and $\tilde{\sigma}^2$ are [73, p. 53]

$$\tilde{\mu} = \bar{\mathbf{x}}$$
$$\tilde{\sigma}^2 = max(0, (\frac{n-1}{n})s^2 - 1),$$

where $\bar{\mathbf{x}}$ is the sample mean and s^2 is the sample variance of past observations. Note that the estimate of the prior is not directly affected by the number of samples used, as would be the case with the posterior distribution.

Mode of posterior

A significantly simpler method [45] of estimating the prior distribution from a given set of data would be to maximize the joint likelihood of $\mathbf{O}$ and θ, i.e. $f(\mathbf{O}, \theta|\phi)$, over θ and ϕ. Here ϕ is the parameter vector of the prior distribution.

Starting with an initial estimate of ϕ, and iteratively using alternate maximization over θ and ϕ, i.e.

$$\theta^{(n)} = \underset{\theta}{argmax}[f(\mathbf{O}, \theta|\phi^{(n)})] \tag{4.35}$$
$$\phi^{(n)} = \underset{\phi}{argmax}[g(\theta^{(n)}|\phi)]. \tag{4.36}$$

we can estimate the prior parameters ϕ and model parameters θ which maximize $f(\mathbf{O}, \theta|\phi)$. The solution of Eq. (4.35) is the mode of the posterior (MAP estimate) for the current prior parameter set. The solution of Eq. (4.36) is the ML estimate of the prior parameters based on the current HMM parameters.

Solving Eq. (4.36) poses two problems:

- ML estimation thereof is not simple as a result of the chosen prior distribution of Section 4.2.
- More prior density parameters must be estimated than for the HMM itself. This is called overparameterization and is a problem.

The overparameterization problem can be overcome by adding certain constraints to the prior parameters, so as to reduce the number of prior parameters to be estimated. The prior family is limited to the posterior density family of the complete data model when no prior information is available. It is then easily shown [45] that the following constraints can be imposed:

$$\delta_{ik} = \nu_{ik} + 1 \tag{4.37}$$

$$n_{ik} = \nu_{ik} + D, \tag{4.38}$$

solving the overparameterization problem to some extent. The parameter δ_{ik} is the mixture weight prior parameter in Eq. (4.14), with n_{ik} and ν_{jk} being prior parameters in Eq. (4.12) for state i and mixture component k.

Note that using Eqs. (4.35) and (4.36) when no prior information is available will result in θ being the mode of the likelihood function (ML estimate) and we therefore set the mode of the prior to be equal to the parameters of a given HMM. Given that the prior

family has been chosen to be the same as that of the complete-data likelihood it makes sense that the mode of the prior will be the ML point estimate.

The prior parameters m_{jk}, τ_{jk} *and* α_{ij} are therefore directly estimated from the ML HMM models, while δ_{ik} and n_{ik} are obtained using Eqs. (4.37) and (4.38). The parameters n_{ik} and ν_{ik} do not determine the mode of the prior distribution, but determine the degree to which the prior is peaked about its mode. The parameter ν_{ik} is therefore a parameter chosen by the user and is typically chosen as a global parameter.

The value of the parameter ν should therefore incorporate *a priori* knowledge about the suitability of the ML model for the task at hand. If the data used to obtain the prior was from the same task as that used to obtain the final MAP point, then we would expect to use a relatively large value for ν. However, if there was a large mismatch between the prior data and data used to obtain the MAP point, then we are not that sure of the prior and a smaller ν would therefore be chosen. An example of where this might occur is in cross-language adaptation, where data from a given language is used to create a prior and the target language's data, along with the prior is used to obtain the MAP point estimate. The influence of ν on the performance of the MAP algorithm will be experimentally be determined in Section 4.6.

Let us investigate the usage of the "mode of posterior" method to determine the prior parameters for the example discussed earlier. The forms of the prior and likelihood functions are identical and therefore there is no problem of overparameterization. If we were to choose the prior mode directly from the ML point, then one would choose $\mu = \bar{\mathbf{x}}$ which is exactly the same as that obtained using the empirical Bayes method. The value of σ^2 is, however, still unknown and it determines the degree to which the prior is peaked about its mode. We could arbitrarily fix it, as we have done with ν above, but we could also choose it to be proportional to the sample variance s^2.

Discussion on prior estimation

If the data samples used were taken from the same source, it would have been better to assume a non-informative (constant) prior distribution and used all the examples to determine the posterior. Note that from Section 4.1.2 we would deduce that if one needed a prior, then the sequential nature of Bayes' theorem tells us that the posterior for the observed data would be the appropriate prior for any subsequently observed data.

If we once again look at the example used in the previous two sections, it is relatively simple to show that the posterior $f(M|\mathbf{x})$ for the given data $\mathbf{x}$ assuming a non-informative prior would be

$$f(M|\mathbf{x}) = \mathcal{N}(\mu, \sigma^2), \tag{4.39}$$

where $\mu = \bar{\mathbf{x}}$ and $\sigma^2 = \frac{1}{N}s^2$. As expected, the posterior becomes more peaked around its mode μ as more data is observed. Following Eq. (4.5), we should use this posterior, as the prior for any new data that is observed.

Neither of the two methods described previously result in a prior which becomes more peaked as the amount of observed data increases. This is a potential problem, as it does not account for the fact that some HMMs (or states or mixtures) will have been observed more often than others. However, the two methods could potentially help the algorithm when there is a reasonable mismatch between prior data and the task specific data.

The sequential nature of Bayes can be used to determine the MAP estimate by using the posterior of the prior data as the prior distribution (Eq. (4.5)). However, there will typically be more prior data than adaptation data, and the prior distribution will therefore tend to dominate in the calculation of the posterior using Eq. (4.5). Any

reasonable mismatch between the prior data and task-specific data will also tend to be a problem. These problems can, however, be addressed by simply weighing the prior distribution (posterior of previously observed data) with a value which is a function of the mismatch (*a-priori* knowledge/belief) and the amount of prior and adaptation data.

4.4 Gradient based MAP

The MAP estimation method proposed by Gauvain and Lee [45], assumes that the prior used is of a specific form. This is potentially a limiting factor in the performance of that MAP algorithm. In this section a gradient based MAP estimation algorithm is developed which does not make assumptions about the form of the prior distribution. This algorithm will, so as to prevent confusion, be referred to as the GMAP algorithm.

The above statement is not entirely true, as a prior of fixed form is used. It is, however, a non-informative prior which is used in the calculation of the new prior, which in turn is then used in the adaptation process. Though, if true prior knowledge about the model or system is available, it can be expressed through this parametric prior.

In the proposed algorithm, the prior will not be estimated at all, but will be implicitly included in the update procedure. It will, however, be far more computationally expensive than the regular MAP algorithm. The improved performance will, however, offset the extra computational difficulties for certain tasks. This adaptation algorithm will probably not find a place in rapid adaptation needed in some speaker adaptation tasks. It should, however, be more than useful in tasks such as cross-language adaptation and some speaker adaptation tasks where training time is not critical.

In the discussion in section 4.3.3, I explained under which circumstances one can use the posterior (or weighted version thereof) of the given "prior" data as the prior distribution. There is no problem with this, except when the same data is used to estimate

the prior and the MAP point of the posterior distribution and should not be done. In Section 4.1.2, it was pointed out that the posterior of a given set of data can be used as the prior for another independent set of data when calculating the posterior of the union of the two sets.

We can rewrite Eq. (4.5) as

$$P(\theta|\mathbf{O}) \propto \left[P(\theta)\prod_{i=1}^{k} L(\theta|\mathbf{O}_i)\right]\prod_{i=k+1}^{n} L(\theta|\mathbf{O}_i) \qquad (k < n). \tag{4.40}$$

The first part in square brackets is the posterior of a set of data $(1 \ldots k)$ and is used as the prior for the remainder of the data. The posterior of this reference or "prior" set, used as the prior in the adaptation framework will tend to dominate Eq. (4.40) when there are more training examples than adaptation examples (i.e. when $k \gg n - k$). MAP adaptation is typically used in situations where this will occur. It therefore makes sense to weigh the "prior" in some way so as to ensure that it does not dominate. In our implementation, the weighting is done as follows

$$P(\theta|\mathbf{O}) \propto \left[P(\theta)\prod_{i=1}^{k} L(\theta|\mathbf{O}_i)\right]^{\varphi}\prod_{i=k+1}^{n} L(\theta|\mathbf{O}_i) \qquad (k < n). \tag{4.41}$$

The value φ has the effect of flattening and widening the prior when $0 \leq \varphi < 1$ and making it more peaked around the mode when $\varphi > 1$. Figure 4.2 presents an example of a Normal distribution with a mean of zero and standard deviation of two, which has been raised to the value of $\varphi = 0.2$ and $\varphi = 2$.

It is convenient at this point to express the posterior in terms of an energy function, which will be optimized to find the MAP estimate. This will become increasingly relevant in Chapter 5.

We can write the posterior probability function as follows

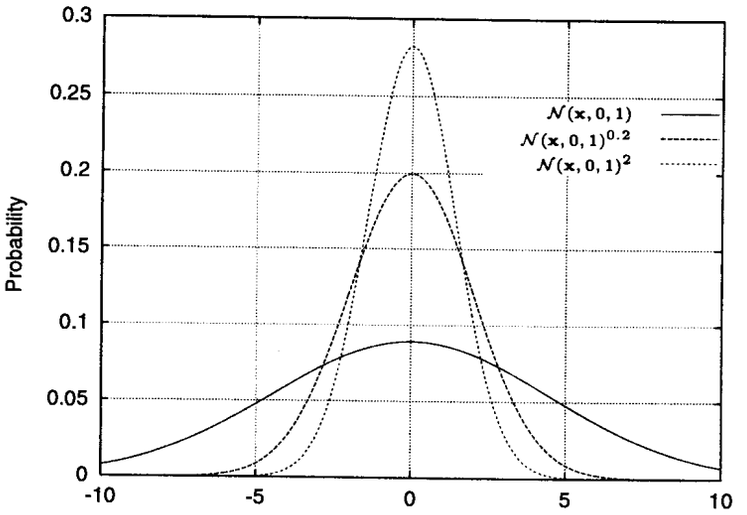


Figure 4.2: Example of the effects of raising a distribution to a power (assuming it is normalized)

$$P(\theta|\mathbf{O}) \propto e^{-E(\theta;\mathbf{O})} \tag{4.42}$$

where $E(\theta;\mathbf{O})$ is the "potential energy" function. Any probability function can be written in this way by defining $E(\theta;\mathbf{O}) = -logP(\theta|\mathbf{O}) - log(Q)$ for a given constant Q ($Q = 1$).

Writing the posterior (Eq. (4.6)) in the above form, and assuming independence of the observations, we get

$$E(\theta;\mathbf{O}) = -log[P(\theta)] - \sum_{i=1}^{n} log[P\mathbf{O}_i|\theta)]. \tag{4.43}$$

The above equation assumes that an informative prior $P(\theta)$ is available. We could again use a parametric prior estimated from the training data as done in the normal MAP approach (Section 4.3). Instead of estimating a prior of fixed form, the prior has been directly included into the posterior calculation (Eq. (4.41)), which can be written as

$$E(\theta;\mathbf{O}) = \varphi\left[-log[P(\theta)] - \sum_{i=1}^{k} log[P(\mathbf{O}_i|\theta)]\right] - \sum_{i=k+1}^{n} log[P(\mathbf{O}_i|\theta)], \tag{4.44}$$

where examples $1,\ldots,k$ are the prior (reference) set and examples $(k+1),\ldots,n$ are the adaptation set.

The steepest descent algorithm [12] can now be used to iteratively estimate the MAP point on the posterior defined in Eq. (4.44), i.e.

$$\theta^{(i)}(n+1) = \theta^{(i)}(n) - \epsilon\frac{\partial E(\theta)}{\partial \theta^{(i)}}, \tag{4.45}$$

where ϵ is the learning rate or step size of the update. The steepest descent algorithm (Eq. (4.45)) is an unconstrained optimization technique and given that certain constraints must be maintained for HMMs, some modifications are required. The next section will investigate the usage of transformations to ensure that the constraints are maintained.

4.4.1 Parameter transformations

Instead of using a complicated constrained steepest descent algorithm, we can, as with minimum classification error (MCE) training (Section 3.2), use transformations to maintain the above constraints during parameter adaptation.

It is necessary to maintain the original HMM constraints, namely

1. $\sum_j a_{ij} = 1$ and $a_{ij} \geq 0$,
2. $\sum_k c_{jk} = 1$ and $c_{jk} \geq 0$, and
3. $\sigma_{jkl} \geq 0$.

The standard parameter transformations given in Table 3.1 are repeated here in Table 4.2 for convenience. These transformations ensure that the unconstrained steepest descent algorithm can be used in the transformed parameter space.

Table 4.2: Parameter transformations used in MCE

Parameter	Transformed parameter	Forward transform	Reverse transform	
a_{ij}	$\tilde{a}_{ij}$	$\tilde{a}_{ij} = \ln(a_{ij})$	$a_{ij} = \frac{e^{\tilde{a}_{ij}}}{\sum_k e^{\tilde{a}_{ij}}}$	Transition probabilities
c_{jk}	$\tilde{c}_{jk}$	$\tilde{c}_{jk} = \ln(c_{jk})$	$c_{jk} = \frac{e^{\tilde{c}_{jk}}}{\sum_k e^{\tilde{c}_{jk}}}$	Mixture weights
μ_{jkl}	$\tilde{\mu}_{jkl}$	$\tilde{\mu}_{jkl} = \frac{\mu_{jkl}}{\sigma_{jkl}}$	$\mu_{jkl} = \sigma_{jkl}\tilde{\mu_{jkl}}$	Gaussian mean
σ_{jkl}	$\tilde{\sigma}_{jkl}$	$\tilde{\sigma}_{jkl} = \ln(\sigma_{jkl})$	$\sigma_{jkl} = e^{\tilde{\sigma}_{jkl}}$	Gaussian std. dev.

These transformations should be kept in mind when calculating both the prior and likelihood derivatives.

4.4.2 Parameter adaptation

The updates for the individual parameter types will now be derived using Eqs. (4.44) and (4.45). Note that the adaptation is done in the transformed parameter space and the parameters are then transformed back to the original parameter space. The derivation of the parameter updates are given in this section. Some of the derivatives will also be used later in Chapter 5.

Gaussian mixture means

The parameter update for the Gaussian mixture mean $\tilde{\mu}_{jkl}$ using the steepest descent algorithm of state j, mixture k and element l is

$$\tilde{\mu}^{(i)}_{jkl}(n+1) = \tilde{\mu}^{(i)}_{jkl}(n) - \epsilon \frac{\partial E(\tilde{\theta}_n)}{\partial \tilde{\mu}^{(i)}_{jkl}}, \tag{4.46}$$

where

$$\begin{aligned}\frac{\partial E(\tilde{\theta}_n)}{\partial \tilde{\mu}_{jkl}^{(i)}} = &-\varphi \frac{\partial \ln[P(\tilde{\theta})]}{\partial \tilde{\mu}_{jkl}^{(i)}} - \varphi \sum_{i=1}^{k} \frac{\partial \ln[P(\mathbf{O}_i|\theta)]}{\partial \tilde{\mu}_{jkl}^{(i)}} \\ &- \sum_{i=k+1}^{n} \frac{\partial \ln[P(\mathbf{O}_i|\theta)]}{\partial \tilde{\mu}_{jkl}^{(i)}}.\end{aligned} \tag{4.47}$$

From Eq. (2.9) we have

$$\frac{\partial \ln[P(\mathbf{O}_i|\theta)]}{\partial \tilde{\mu}_{jkl}^{(i)}} = \sum_{t=1}^{T} \delta(\bar{q}_t - j) \frac{\partial \ln(b_j^{(i)}(\mathbf{o}_t))}{\partial \tilde{\mu}_{jkl}^{(i)}}, \tag{4.48}$$

where $\delta()$ is the Kronecker delta function and $b_j^{(i)}(\mathbf{o}_t)$ is the observation probability (Eq. (2.5)) of state j of HMM i at time t. Assuming a diagonal covariance matrix, we get

$$\begin{aligned}\frac{\partial \ln(b_j^{(i)}(\mathbf{o}_t))}{\partial \tilde{\mu}_{jkl}^{(i)}} =& c_{jk}^{(i)}(2\pi)^{-d/2}|\Sigma_{jk}^{(i)}|^{-1/2}(b_j^{(i)}(\mathbf{o}_t))^{-1}\left(\frac{\mathbf{o}_{tl}}{\sigma_{jkl}^{(i)}} - \tilde{\mu}_{jkl}^{(i)}\right) \\ & e^{-\frac{1}{2}\sum_{l=1}^{D}\left(\frac{\mathbf{o}_{tl}}{\sigma_{jkl}^{(i)}} - \tilde{\mu}_{jkl}^{(i)}\right)^2} \\ =& \frac{c_{jk}\mathcal{N}(\mathbf{o}_t; \mu_{jk}, \Sigma_{jk})}{b_j^{(i)}(\mathbf{o}_t)} \cdot \left(\frac{\mathbf{o}_{tl} - \mu_{jkl}}{\sigma_{jkl}}\right).\end{aligned} \tag{4.49}$$

Note that the derivatives are sometimes written in terms of the original parameter and not in terms of the transformed parameter. The partial derivative of $\ln[P(\tilde{\theta})]$ with respect to $\tilde{\mu}_{jkl}^{(i)}$ can be obtained from Eq. (4.13),

$$\begin{aligned}\frac{\partial g(\hat{\theta})}{\partial \tilde{\mu}_{jkl}^{(i)}} =& -\frac{1}{2}\nu\sigma_{jkl}^{-2} \cdot 2(\tilde{\mu}_{jkl}\sigma_{jkl} - M_{jkl}) \cdot \sigma_{jkl} \\ =& -\nu\sigma_{jkl}^{-1}(\mu_{jkl} - M_{jkl}).\end{aligned} \tag{4.50}$$

Having updated the transformed mean using Eq. (4.46), the correct mean can be found using the inverse transformation $\mu_{jkl}^{(i)}(n+1) = \tilde{\mu}_{jkl}^{(i)}(n)\sigma_{jkl}^{(i)}(n+1)$ from Table 4.2.

Gaussian mixture variance

The steepest descent update for the Gaussian mixture variance $\tilde{\sigma}_{jkl}^2$ for state j, mixture k and element l is

$$\tilde{\sigma}_{jkl}^{(i)}(n+1) = \tilde{\sigma}_{jkl}^{(i)}(n) - \epsilon \frac{\partial E(\tilde{\theta}_n)}{\partial \tilde{\sigma}_{jkl}^{(i)}} \tag{4.51}$$

where

$$\begin{aligned} \frac{\partial E(\tilde{\theta}_n)}{\partial \tilde{\sigma}_{jkl}^{(i)}} = &- \varphi \frac{\partial \ln[P(\tilde{\theta})]}{\partial \tilde{\sigma}_{jkl}^{(i)}} - \varphi \sum_{i=1}^{k} \frac{\partial \ln[P(\mathbf{O}_i|\theta)]}{\partial \tilde{\sigma}_{jkl}^{(i)}} \\ &- \sum_{i=k+1}^{n} \frac{\partial \ln[P(\mathbf{O}_i|\theta)]}{\partial \tilde{\sigma}_{jkl}^{(i)}}. \end{aligned} \tag{4.52}$$

From Eq. (2.9) we can again write

$$\frac{\partial \ln[P(\mathbf{O}_i|\theta)]}{\partial \tilde{\sigma}_{jkl}^{(i)}} = \sum_{t=1}^{T} \delta(\bar{q}_t - j) \frac{\partial \ln(b_j^{(i)}(\mathbf{o}_t))}{\partial \tilde{\sigma}_{jkl}^{(i)}}, \tag{4.53}$$

where (from Eq. (2.5))

$$\begin{aligned}\frac{\partial \ln(b_j^{(i)}(\mathbf{o}_t))}{\partial \tilde{\sigma}_{jkl}^{(i)}} =& c_{jk}^{(i)}(2\pi)^{-d/2}|\Sigma_{jk}^{(i)}|^{-1/2}(b_j^{(i)}(\mathbf{o}_t))^{-1} \\ & \left[\left(\frac{\mathbf{o}_{tl}-\mu_{jkl}^{(i)}}{\sigma_{jkl}^{(i)}}\right)^2 - 1\right] e^{-\frac{1}{2}\sum_{l=1}^{D}\left(\frac{\mathbf{o}_{tl}-\mu_{jkl}^{(i)}}{\sigma_{jkl}^{(i)}}\right)^2} \\ =& \frac{c_{jk}\mathcal{N}(\mathbf{o}_t;\mu_{jk},\Sigma_{jk})}{b_j^{(i)}(\mathbf{o}_t)} \cdot \left[\left(\frac{\mathbf{o}_{tl}-\mu_{jkl}^{(i)}}{\sigma_{jkl}^{(i)}}\right)^2 - 1\right].\end{aligned} \tag{4.54}$$

The partial derivative of $\ln[P(\tilde{\theta})]$ with respect to $\tilde{\sigma}_{jkl}^{(i)}$ can again be obtained from Eq. (4.13),

$$\begin{aligned}\frac{\partial g(\hat{\theta})}{\partial \tilde{\sigma}_{jkl}^{(i)}} &= \frac{\partial g(\hat{\theta})}{\partial \sigma_{jkl}^{(i)}}\frac{\partial \sigma_{jkl}^{(i)}}{\partial \tilde{\sigma}_{jkl}^{(i)}} = \left[\nu\frac{(\mu_{jkl}-M_{jkl})^2}{\sigma_{jkl}^3} - \frac{(n-D)}{\sigma_{jkl}} + \frac{\tau_{jkl}}{\sigma_{jkl}^3}\right]\cdot(e_{jkl}^{\tilde{\sigma}}) \\ &= \nu\frac{(\mu_{jkl}-M_{jkl})^2}{\sigma_{jkl}^2} - (n-D) + \frac{\tau_{jkl}}{\sigma_{jkl}^2}.\end{aligned} \tag{4.55}$$

After the update in the transformed space the new standard deviation can be found using $\sigma_{jkl}^{(i)}(n+1) = e^{\tilde{\sigma}_{jkl}(n+1)}$.

Gaussian mixture weights

The steepest descent update for the Gaussian mixture weights is

$$\tilde{c}_{jk}^{(i)}(n+1) = \tilde{c}_{jk}^{(i)}(n) - \epsilon\frac{\partial E(\tilde{\theta}_n)}{\partial \tilde{c}_{jk}^{(i)}} \tag{4.56}$$

where

$$\frac{\partial E(\tilde{\theta}_n)}{\partial \tilde{c}_{jk}^{(i)}} = -\varphi \frac{\partial \ln[P(\tilde{\theta})]}{\partial \tilde{c}_{jk}^{(i)}} - \varphi \sum_{i=1}^{k} \frac{\partial \ln[P(\mathbf{O}_i|\theta)]}{\partial \tilde{c}_{jk}^{(i)}} - \sum_{i=k+1}^{n} \frac{\partial \ln[P(\mathbf{O}_i|\theta)]}{\partial \tilde{c}_{jk}^{(i)}}. \tag{4.57}$$

The partial derivative of Eq. (2.9) with respect to $\tilde{c}_{jk}^{(i)}$ is

$$\frac{\partial \ln[P(\mathbf{O}_i|\theta)]}{\partial \tilde{c}_{jk}^{(i)}} = \sum_{t=1}^{T} \delta(\bar{q}_t - j) \frac{\partial \ln(b_j^{(i)}(\mathbf{o}_t))}{\partial \tilde{c}_{jk}^{(i)}}, \tag{4.58}$$

where

$$\begin{aligned} \frac{\partial \ln(b_j^{(i)}(\mathbf{o}_t))}{\partial \tilde{c}_{jk}^{(i)}} =& c_{jk}^{(i)} \left[(2\pi)^{-d/2} |\sigma_{jk}^{(i)}|^{-1/2} (b_j^{(i)}(\mathbf{o}_t))^{-1} \right. \\ & \left. e^{-\frac{1}{2}\sum_{l=1}^{D} \left(\frac{o_{tl} - \mu_{jkl}^{(i)}}{\sigma_{jkl}^{(i)}} \right)^2} - 1 \right] \\ =& c_{jk}^{(i)} \left[\frac{\mathcal{N}(\mathbf{o}_t; \mu_{jk}, \Sigma_{jk})}{b_j^{(i)}(\mathbf{o}_t)} - 1 \right]. \end{aligned} \tag{4.59}$$

The prior distribution for the mixture weights is a Dirichlet distribution and we therefore obtain the partial derivative of $\ln[P(\tilde{\theta})]$ with respect to $\tilde{c}_{jk}^{(i)}$ from Eq. (4.14).

$$\begin{aligned} \frac{\partial \ln[P(\tilde{\theta})]}{\partial \tilde{c}_{jk}^{(i)}} =& \frac{\partial g(\hat{\theta})}{\partial c_{jk}^{(i)}} \frac{\partial c_{jk}^{(i)}}{\partial \tilde{c}_{jk}^{(i)}} \\ =& \sum_m \left[(\delta_{jm} - 1) \frac{\partial \ln(c_{jm})}{\partial c_{jm}} \cdot \frac{\partial c_{jm}^{(i)}}{\partial \tilde{c}_{jk}^{(i)}} \right] \\ =& \sum_m \left[(\delta_{jm} - 1) \cdot c_{jm}^{-1} \cdot (\delta(k-m) c_{jm} - c_{jk} c_{jm}) \right] \\ =& (\delta_{jk} - 1) - \sum_m (\delta_{jm} - 1) c_{jm}. \end{aligned} \tag{4.60}$$

The new weight can be found by transforming back to the true weight space, using $c_{jk} = \frac{e^{\tilde{c}_{jk}}}{\sum_k e^{\tilde{c}_{jk}}}$.

Transition probabilities

The steepest descent update for the transition probabilities is

$$\tilde{a}_{ij}^{(i)}(n+1) = \tilde{a}_{ij}^{(i)}(n) - \epsilon \frac{\partial E(\tilde{\theta}_n)}{\partial \tilde{a}_{ij}^{(i)}} \tag{4.61}$$

where

$$\begin{aligned} \frac{\partial E(\tilde{\theta}_n)}{\partial \tilde{a}_{ij}^{(i)}} = &- \varphi \frac{\partial \ln[P(\tilde{\theta})]}{\partial \tilde{a}_{ij}^{(i)}} - \varphi \sum_{i=1}^{k} \frac{\partial \ln[P(\mathbf{O}_i|\theta)]}{\partial \tilde{a}_{ij}^{(i)}} \\ &- \sum_{i=k+1}^{n} \frac{\partial \ln[P(\mathbf{O}_i|\theta)]}{\partial \tilde{a}_{ij}^{(i)}}. \end{aligned} \tag{4.62}$$

The partial derivative of Eq. (2.9) with respect to $\tilde{a}_{ij}^{(i)}$ is

$$\frac{\partial \ln[P(\mathbf{O}_i)]}{\partial \tilde{a}_{ij}^{(i)}} = \sum_{t=1}^{T} \sum_{s} \delta(\bar{q}_{t-1} - i)\delta(\bar{q}_t - s)\Big[\delta(j-s) - a_{ij}\Big]. \tag{4.63}$$

The prior distribution for the transition probabilities is a Dirichlet distribution and we therefore obtain the partial derivative of $\ln[P(\tilde{\theta})]$ with respect to $\tilde{c}_{jk}^{(i)}$ from Eq. (4.15).

$$\begin{aligned}\frac{\partial \ln[P(\tilde{\theta})]}{\partial \tilde{a}_{ij}^{(i)}} =& \frac{\partial g(\hat{\theta})}{\partial a_{ij}^{(i)}} \frac{\partial a_{ij}^{(i)}}{\partial \tilde{a}_{ij}^{(i)}} \\ =& \sum_k \left[(\alpha_{ik} - 1) \frac{\partial \ln(a_{ik})}{\partial a_{ij}} \cdot \frac{\partial a_{ij}^{(i)}}{\partial \tilde{a}_{ij}^{(i)}} \right] \\ =& \sum_k \left[(\alpha_{ik} - 1) \cdot a_{ik}^{-1} \cdot (\delta(j-k) a_{ik} - a_{ij} a_{ik}) \right] \\ =& (\alpha_{ij} - 1) - \sum_k (\alpha_{ik} - 1) a_{ik}.\end{aligned} \tag{4.64}$$

Finally, the updated transition probability is found by transforming back using $a_{ij} = \frac{e^{\tilde{a}_{ij}}}{\sum_k e^{\tilde{a}_{ij}}}$.

4.4.3 Initial prior

Although the GMAP algorithm has been developed such that an initial parametric prior can be used (to incorporate true *a-priori* information), it is unlikely that any such information will exist. This algorithm will therefore often be used with a non-informative initial prior. This results in the prior gradient being zero, with no effect on the resultant update.

4.5 MAPMCE

The term MAP-MCE is sometimes used to refer to the usage of the MCE algorithm, but starting at the MAP point. This section describes an entirely different technique, which has been named MAPMCE in this work for the reason that it estimates the MAP point of the posterior of the probability of choosing the correct class (MCE), i.e. it optimizes the classification error (as criterion function) with respect to the parameters while incorporating a prior in the formulation.

In Section 3.2 it was mentioned that the MCE loss function is a reasonable estimate of the error probability. Given that this is the case, we can estimate the MAP point for the posterior distribution of the probability of choosing the correct class, namely

$$P(\theta, C_i|\mathbf{O}) \propto P(C_i|\theta, \mathbf{O})P(\theta), \tag{4.65}$$

where C_i is the correct class and $P(C_i|\theta, \mathbf{O})$ is the probability of choosing the correct class C_i given the current model parameters θ and the observations $\mathbf{O}$.

If the MCE loss function is a good estimate of the error probability, then we can write

$$P(C_i|\theta, \mathbf{O}) \approx 1 - l(\mathbf{O}; \theta), \tag{4.66}$$

where $l(\mathbf{O}; \theta)$ is the loss function defined in Eq. (3.15).

Rewriting Eq. (4.41) for the posterior in Eq. (4.65) and using the MCE loss function, $P(C_i|\theta, \mathbf{O})$ in Eq. (4.66) instead of the likelihood function $L(\theta|\mathbf{O})$ (or $P(\mathbf{O}|\theta)$), we get

$$P(\theta|\mathbf{O}) \propto \left[P(\theta)\prod_{i=1}^{k}(1 - l(\mathbf{O}_i; \theta))\right]^{\varphi} \prod_{i=k+1}^{n}(1 - l(\mathbf{O}_i; \theta)) \qquad (k < n). \tag{4.67}$$

As mentioned in Section 4.4.3, there is little chance of a parametric prior being available for the gradient based MAP algorithm and this is also the case for the MAPMCE algorithm. The rest of this procedure will therefore be developed without the parametric prior $P(\theta)$.

Following the same reasoning as in Section 4.4, we can express Eq. (4.67) in terms of an energy function,

$$E(\theta;\mathbf{O}) = -\varphi\sum_{i=1}^{k}\ln[(1-l(\mathbf{O}_i;\theta))] - \sum_{i=k+1}^{n}\ln[(1-l(\mathbf{O}_i;\theta))]. \tag{4.68}$$

Unfortunately the gradient ($\frac{\partial E(\theta)}{\partial\theta}$) for Eq. (4.68) is not finite, with $\frac{\partial E(\theta)}{\partial\theta} = \infty$ for any $l(\mathbf{O}_i;\theta) = 1$.

A different estimate of the probability $P(C_i|\theta,\mathbf{O})$ has therefore been chosen, namely

$$P(C_i|\theta,\mathbf{O}) \approx e^{-l(\mathbf{O}_i;\theta)}. \tag{4.69}$$

Although this is not an entirely accurate estimate of the probability, it results in the following, more convenient, energy function

$$E(\theta;\mathbf{O}) = \varphi\sum_{i=1}^{k}l(\mathbf{O}_i;\theta) + \sum_{i=k+1}^{n}l(\mathbf{O}_i;\theta), \tag{4.70}$$

where $l(\mathbf{O}_i;\theta)$ is the MCE loss function (Eq. (3.15)) for observation i. Equation (4.70) is intuitively pleasing as it is simply a weighted version of the standard MCE algorithm in the sense that the "prior" terms ($i = 1\ldots k$) are weighted by φ. An error in the "prior" or reference set will therefore not be penalized as heavily as an error in the adaptation set ($0 \leq \varphi \leq 1$). The algorithm will therefore try to minimize errors in both sets, but will place more emphasis on errors that occur in the adaptation set.

The implementation of this algorithm, requires the following simple modification to the standard MCE algorithm which was described in Chapter 3: if an observation is in the "prior" dataset, then the gradient with respect to the model parameters ($\nabla l(\mathbf{O};\theta)$) used in the GPD update (Eq. (3.18)) must be weighed with φ.

4.6 Experiments

The goal of this section is to experimentally compare the three algorithms discussed earlier in this chapter (MAP, GMAP and MAPMCE) in conditions where limited training data is available, with a reasonable amount of non-task-specific data available for adaptation purposes.

Before continuing, it is necessary to explain the convention I have used to describe (or label) the algorithms used:

- MAP - A MAP algorithm labelled as "MAP T_X (T_Y)" uses the dataset T_X as its adaptation set and the ML model created using T_Y to determine the prior distribution. Here, Eqs. (4.22) through (4.26) are used.
- GMAP - A GMAP algorithm labelled as "GMAP T_X (T_Y)" uses the dataset T_X as its adaptation set and the ML model created using T_Y as a starting point. Note that the prior dataset is not included in the description as it is a constant for each experiment. Here, Eqs. (4.46) through (4.64) are used.
- MAPMCE - A MAPMCE algorithm labelled as "MAPMCE T_X (T_Y)" uses the dataset T_X as its adaptation set and the ML model created using T_Y as a starting point. As with GMAP, the prior dataset is not included in the description thereof. Here, Eq. (4.70) is implemented using Eq. (3.25) through Eq. (3.46).

The number of iterations used for each procedure differed. When using MAP, 10 iteration typically proved to be sufficient with the testing set performance converging at or before 10 iterations. GMAP required between 10 and 30 iterations for the testing set performance to converge, while the MAPMCE algorithm also required between 10 and 30 iterations to attain peak testing set performance. The number of iterations required for the gradient-based algorithms (GMAP and MAPMCE) is dependent on the step size ϵ, the size of the datasets and the weighting factor φ.

In situations where HMMs of differing complexity are used, the number of states and mixtures will be included in the description. For example, the algorithm description "MAP 3,5 T_X (T_Y)" refers to the MAP algorithm using a 3 state, 5 mixture HMM.

4.6.1 SUNSpeech

This section compares the three adaptation algorithms within a language adaptation framework. The SUNSpeech dataset described in Section 2.4 is used for this purpose. The subsets of the dataset described in Section 2.4.3 are used for this experiment.

Table 4.3 presents the Afrikaans test set accuracy of the base system (described in Section 2.1) for the different training sets. As expected, the performance of the system trained using the English subset (31.9% and 33.9% for 5 and 10 mixtures respectively) is worse than that trained using either the full Afrikaans training set (48.6% and 51.5%) or the reduced Afrikaans training subset (42.5% and 41.2%). Training using combined English and Afrikaans sets produces relatively poor results as compared to only using the associated Afrikaans set.

Table 4.3: Base system results for SUNSpeech Afrikaans test set

Training set	Mixtures	
	5	10
English (E)	31.9%	33.9%
Afrikaans train (A)	48.6%	51.5%
Afrikaans train + English ($A + E$)	37.9%	41.4%
Afrikaans adapt (A_S)	42.5%	41.2%
Afrikaans adapt + English ($A_S + E$)	34.2%	37.6%

Using 10 mixture components as opposed to 5 improves results, except when using the reduced Afrikaans training set where a reduction in performance occurs. This phenomenon can be ascribed to the bias/variance problem discussed in Section 2.2.1.

The more complex 10 mixture HMM has a lower bias, and will therefore work better in situations where there is sufficient data available. In situations where there is little data available, the variance term becomes dominant and so the less complex model (5 mixture HMM) works best.

The recognition results appear relatively low, as compared to that obtained for continuous recognition on other continuous phoneme recognition tasks, such as TIMIT. The recognition performance of the same system using the TIMIT database is 57.0% and 60.0% for 3 state models using 5 and 10 mixtures respectively. There are several reasons that could account for the disparity in recognition performance, including:

- There are a total of 59 phonetic categories (including *silence* and *unknown*) used in SUNSpeech. This is as compared to, for example, TIMIT where there are 39 phonetic categories. The task is therefore somewhat more complex.
- The SUNSpeech dataset labels include an *unknown* class, which accounts for 1.9% of the labels in the test set. The *unknown* class is not modeled, this will result in at best a 1.9% poorer error rate. The *unknown* class includes any sounds or speech that cannot be included in any of the other 58 phonetic categories.

MAP

This section presents the results obtained for the MAP algorithm described in Section 4.3 using the SUNSpeech dataset. The adaptation set and prior are an integral part of the MAP algorithm and will therefore be included in any description thereof.

Figure 4.3 presents the MAP adaptation results for a 3 state, 5 mixture system when only the reduced Afrikaans training set (A_S) is available. The line labelled "ML A_S" gives the base maximum likelihood performance (42.5%) of a 3 state, 5 mixture model trained on the Afrikaans adaptation set only. The MAP algorithms "MAP A_S (E)"

and "$MAPA_S(A_S + E)$" use a prior created using the English training set (E) and a prior made using the union of the English and Afrikaans adaptation sets $(A_S + E)$.

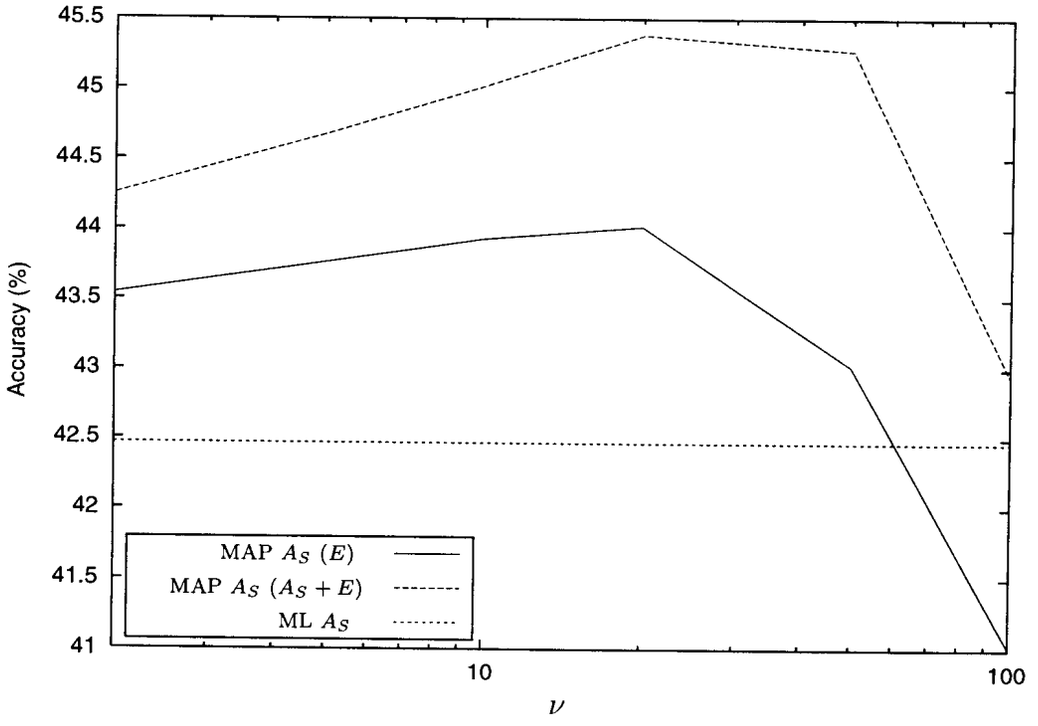


Figure 4.3: MAP adaptation results plotted versus the prior parameter ν (Eq. (4.12)) for adaptation set A_S using an English prior (E) and a prior created using pooled English and Afrikaans adaptation set $(A_S + E)$ for a 3 state, 5 mixture model.

Peak performance of 45.4% phoneme accuracy is attained using the pooled set (A_S+E) to create the prior (5% relative improvement in error rate). The MAP algorithm using a prior created using the English training set only performed considerably worse, reaching a peak performance of 44.0% (2.6% relative improvement in error rate).

Both implementations attain peak performance with $\nu = 20$. As expected, the performance of the MAP algorithm is highly dependent on the value chosen for the parameter ν. If ν is too small, the prior would not be informative enough and little prior information is therefore obtained from the prior training set. A large value of ν, on the other hand, would result in a prior that is too restrictive, thereby keeping the resultant model very close to the model trained on the prior training set. The recognition accuracy is therefore expected to tend towards that of an ML model trained using the only

the adaptation set (ML A_S) for very small values of ν. Conversely, the recognition accuracy is expected to tend towards that of the ML model trained using the prior training set (E) for large values of ν.

There is, however, an optimal value for ν between these two extremes. This occurs when information from both sets is being used optimally.

Figure 4.4 shows the the MAP adaptation results for a 3 state, 5 mixture HMM system when the full Afrikaans training set (A) is used as the adaptation set. The results for the MAP algorithm using both the English set prior and a pooled prior created using both the English set and full Afrikaans training set ($A + E$) are compared.

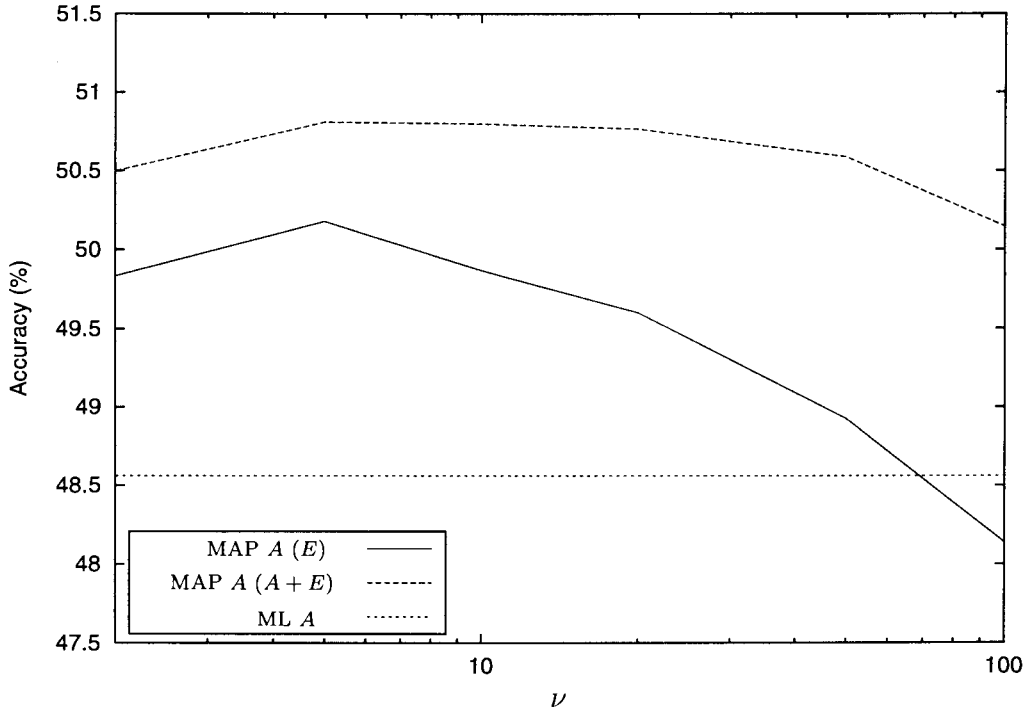


Figure 4.4: MAP adaptation results using full Afrikaans set

The recognition accuracy for the base ML system (ML A) is 48.6%. The MAP algorithm using the pooled prior ($A + E$) with peak performance of 50.8% (4.3% relative improvement of error rate), once again performed better than the MAP algorithm using only the English prior (E) with a peak performance of 50.2% (3.1% relative

improvement of error rate).

Here, peak performance was attained for $\nu = 5$ for both algorithms, which is less than that required when the smaller Afrikaans training set is used. As the amount of task-specific training data increases, one would expect the optimal value of ν to decrease, as has been demonstrated in this experiment.

One would also expect the MAP algorithm to become less useful as the amount of task-specific training data increases. This is evident here, with the relative improvement in performance (4.3%) being less than that obtained for the small Afrikaans training set (5%).

The results for a 3 state, 10 mixture HMM model experiment using only the reduced Afrikaans set (A_S) are shown in Figure 4.5 (the priors are either E or $A_S + E$). Maximum recognition accuracy of 44.8% is attained using the pooled prior, while the MAP algorithm using only the English prior again does not work as well managing a maximum of 43.8%.

The results are very similar to that obtained for the less complex 3 state, 5 mixture HMM. It is, however, apparent that the absolute performance is worse than that of the the 3 state, 5 mixture model (44.8% versus 45.4%).

Table 4.4 summarizes the best accuracies attained using the MAP algorithm for the different configurations. Note that the 3 state, 10 mixture model is consistently worse than the 5 mixture variant when the small Afrikaans training set is used. It is also noticeable that the effect of the MAP algorithm is reduced as the amount of Afrikaans data (adaptation data) is increased.

Importantly, the MAP algorithm was not able to improve on the ML results for the 3 state 10 mixture model, when using the English prior with the full Afrikaans training set.

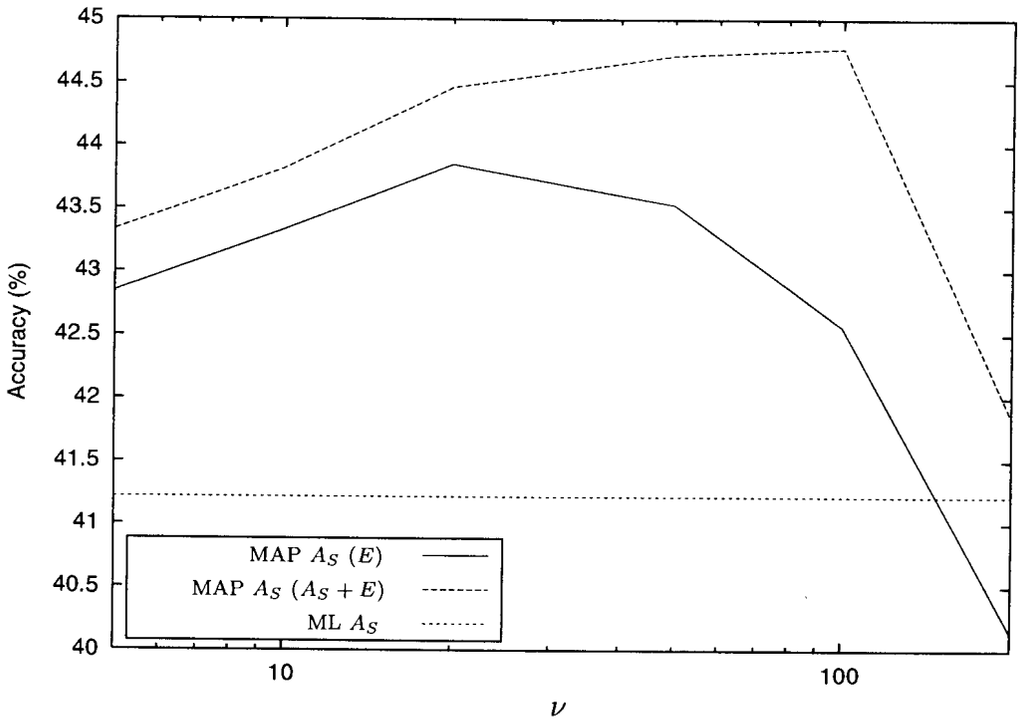


Figure 4.5: MAP adaptation results for a 3 state, 10 mixture HMM using the small Afrikaans training set

GMAP

The gradient based MAP algorithm presented in Section 4.4 is tested using the SUN-Speech dataset here. Incorporating the adaptation data in both the prior and adaptation datasets results in the adaptation data (Eq. (4.44)) having an effective weight of $(1 + \varphi)$ with the prior data having a weight of φ. The same result can be accomplished, without incorporating the adaptation data in the prior data, by choosing a new weighting factor $\varphi' = \frac{\varphi}{(1+\varphi)}$. There is therefore no point in pooling the English and Afrikaans datasets and using the resultant dataset as a prior dataset, as was done for the standard MAP algorithm. As a result, the English dataset is used as the prior dataset for all experiments conducted in this section.

As a result of using the steepest descent algorithm, one would expect this algorithm to easily fall into local minima. The starting point, should therefore influence the

Table 4.4: Summary of the MAP accuracy results obtained for the SUNSpeech database. Relative improvement in error rate over baseline performance is given in brackets.

Description	Mixtures	
	5	10
Baseline ML (A_S)	42.5% (0.0%)	41.2% (0.0%)
MAP A_S (E)	44.0% (2.6%)	43.9% (4.6%)
MAP A_S ($A_S + E$)	45.4% (5.0%)	44.8% (6.1%)
Baseline ML (A)	48.6% (0.0%)	51.5% (0.0%)
MAP A (E)	50.2% (3.1%)	51.25% (-0.5%)
MAP A ($A + E$)	51.1% (4.9%)	52.4% (1.9%)

performance of the algorithm to some degree. Figure 4.6 presents the results using the gradient based MAP algorithm for a 3 state, 5 mixture HMM, starting at (a) the English ML model, (b) the ML model for the small Afrikaans training set and (c) the best MAP estimate from the previous section. The baseline (ML) accuracies for the large dataset (ML A) and small training set (ML A_S) are also plotted.

Starting the algorithm using the English ML model, results in relatively poor accuracy (peak accuracy of 35%), considerably worse than that of the ML model of the small Afrikaans training set. Using the ML model of the small Afrikaans training set and the best MAP estimate (which produced an accuracy of 45.4%) as starting points, produces good improvements in accuracy relative to that of the ML model (peak accuracy of 46.0% and 47.1% respectively).

Figure 4.7 shows the performance of the gradient based MAP algorithm for both 5 and 10 mixture HMMs. The starting point was chosen as the best MAP estimate for all algorithms presented in this graphic.

It is important to note that here, the performance of the 10 mixture HMM is better than that of the 5 mixture HMM. This is, as opposed to that of standard MAP and the ML algorithms, where the 3 state 10 mixture HMM performed slightly worse.

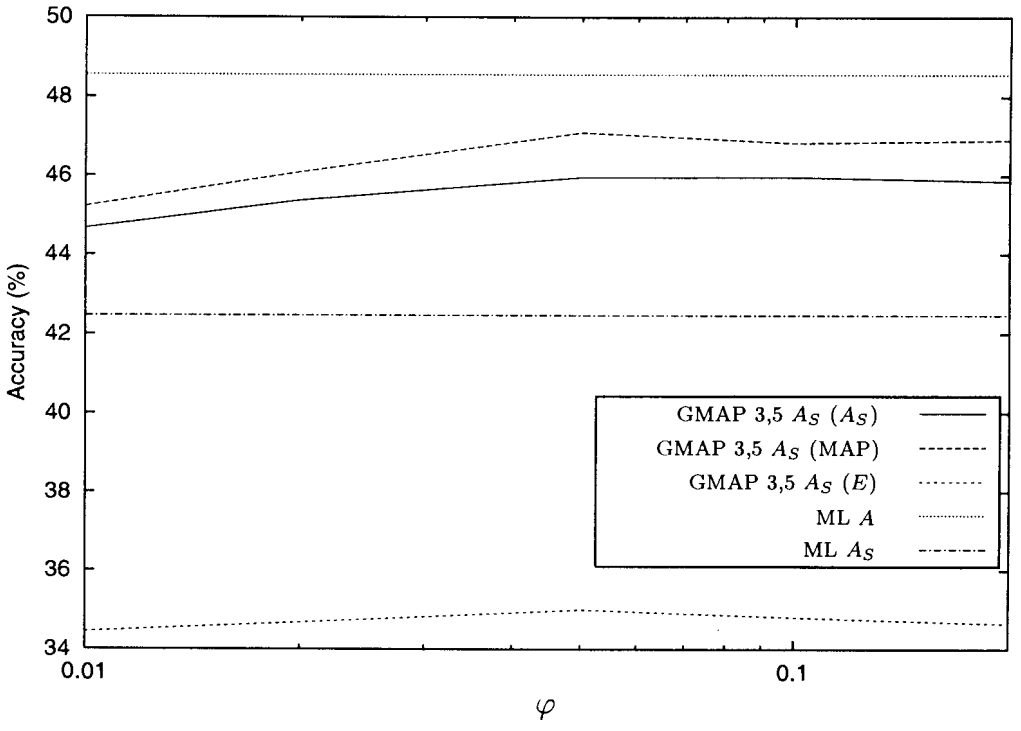


Figure 4.6: Gradient MAP adaptation results for the small Afrikaans training set (A_S)

The weighting factor φ (Eq. (4.44)) has much the same effect as the parameter ν (Eq. (4.12)) has in the standard MAP algorithm. Although it is not easily observable from the results presented here, this algorithm exhibits asymptotic behaviour in the extremes of φ. For $\varphi = 0$ the training examples in the prior set would be ignored and the algorithm would become an ML algorithm. A value of 1 would weight prior and adaptation examples equally and the algorithm becomes an ML algorithm trained on the pooled dataset.

Table 4.5 summarizes the results obtained when using the GMAP algorithm. The GMAP algorithm produces significant improvements on the baseline performance when the small Afrikaans training set is available. Note that this is not the case when the English set is used as starting point, and the starting point should therefore be chosen carefully. Smaller, but significant improvements are realized when the full Afrikaans training set is used.

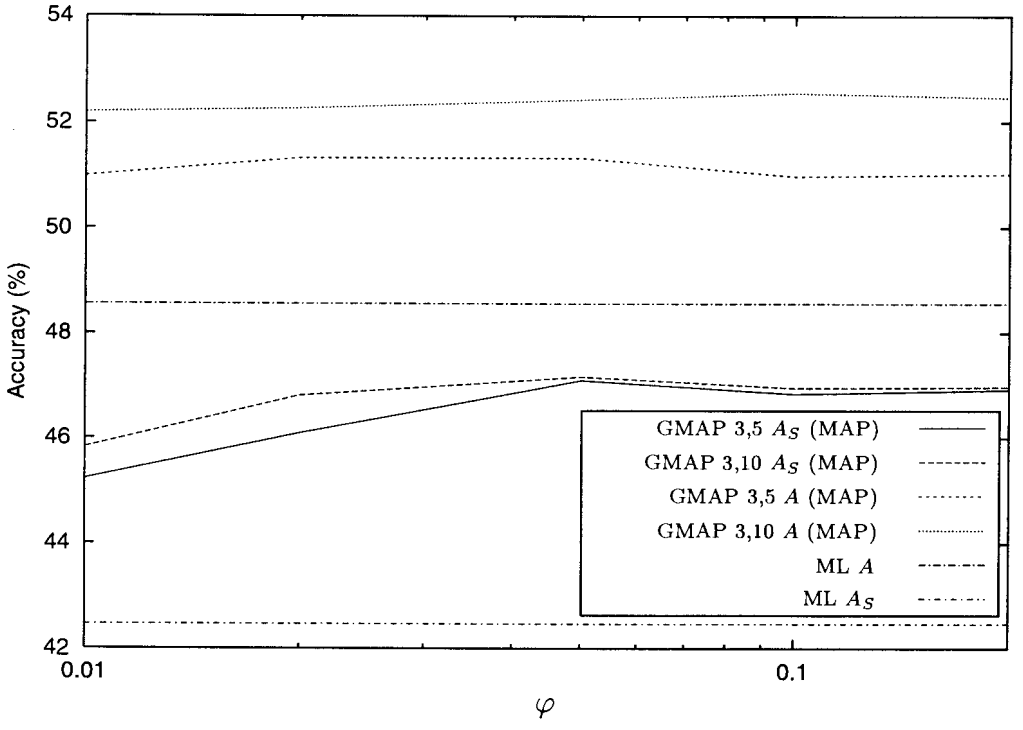


Figure 4.7: Gradient MAP adaptation results for 5 and 10 mixture HMMs

MAPMCE

The MAPMCE algorithm presented in Section 4.5 is tested using the SUNSpeech dataset here. The English dataset is used as the prior dataset for all experiments conducted in this section.

In Chapter 3 it was argued that it was not necessary to use a sigmoid (or smoothed zero-one) loss function and results indicated that there was little to be gained from the use thereof. It is therefore necessary to again investigate the effect of a smoothed zero-one loss function within the MAPMCE framework.

Figure 4.8 presents the results using the MAPMCE algorithm for a 3 state 5 mixture model using the small Afrikaans training set, starting from the English ML model. The results for both sigmoid and no-sigmoid loss functions are shown. The baseline performance, namely that of the ML model trained using the small Afrikaans set, is

Table 4.5: Summary of the accuracy results obtained using the GMAP algorithm

Description	Mixtures	
	5	10
Baseline ML (A_S)	42.5%	41.2%
GMAP A_S (E)	35.0%	34.9%
GMAP A_S (A_S)	46.0%	46.2%
GMAP A_S (MAP)	47.1%	47.2%
Baseline ML (A)	48.6%	51.5%
GMAP A (MAP)	51.3%	52.5%

shown.

The MAPMCE algorithm using a sigmoid loss function attains a peak accuracy of 32.4% for small values of φ. This is not sufficient to improve on the baseline performance of the ML model. The MAPMCE algorithm which does not make use of a sigmoid loss function, however, performs considerably better and manages to improve on the baseline performance. A peak accuracy of 43.5% is attained, resulting in a relative improvement in error rate of 1.8% as compared to that of the baseline ML system.

These results confirm the discussion in Chapter 3, showing that the sigmoid function merely serves to introduce additional local minima, and therefore hinders the algorithm from reaching a meaningful local minimum. All subsequent experiments using the MAPMCE algorithm will therefore not use a smoothed zero-one loss function.

Figure 4.9 shows the results of the MAPMCE algorithm for various configurations. The ML results for a 3 state, 5 mixture HMM are shown for both the full (ML 3,5 A) and reduced Afrikaans (ML 3,5 A_S) training sets for reference purposes. The starting point for all the results presented here is the best associated MAP estimate.

It is evident from Figure 4.9 that although the algorithm does exhibit a degree of

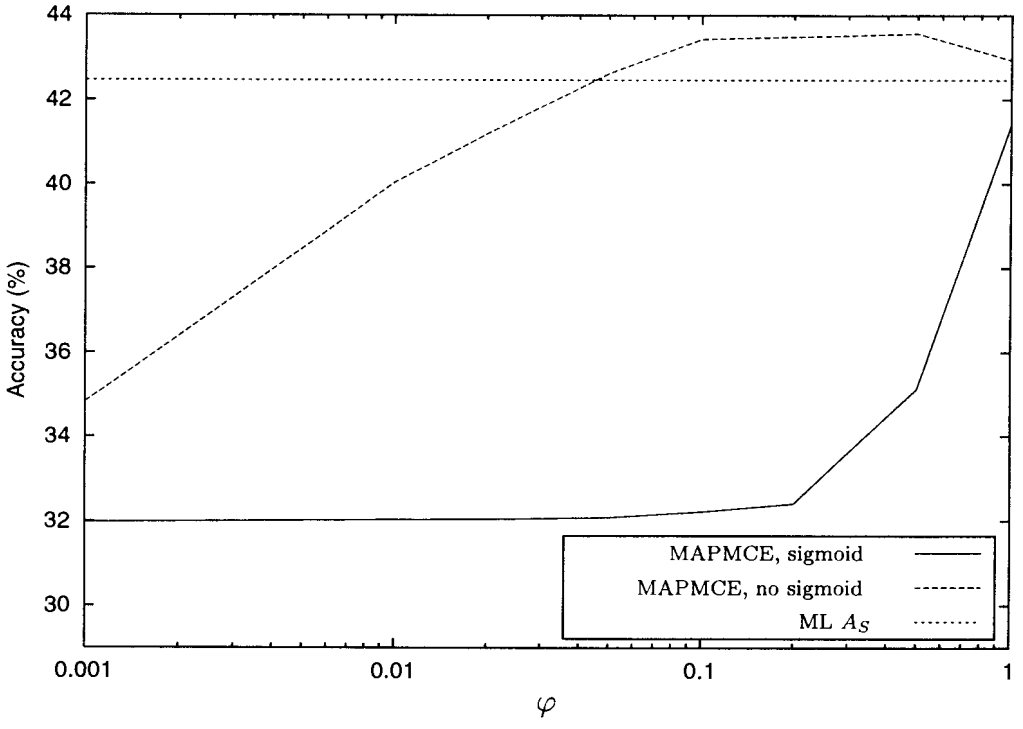


Figure 4.8: MAPMCE adaptation results for the small Afrikaans training set when using the English ML model as starting point

dependence on the weighting parameter φ, it is not overly sensitive to it.

We would prefer an algorithm to perform better with more complex models, with performance increasing as the model complexity increased. Unfortunately, here the algorithm performs best for the reduced dataset, when using the less complex 5 mixture per state HMM. As expected, the opposite is true when the full training set is used, resulting in the more complex model performing best.

Table 4.6 summarizes the results obtained for the MAPMCE algorithm when using the SUNSpeech dataset. Here, unlike the GMAP algorithm an improvement in performance is realized when using the English ML model as starting point, although this is not the case when using a sigmoid loss function. The MAP point estimate was once again found to be the best starting point for the MAPMCE algorithm. Reasonable improvements are attained for both the small and full Afrikaans training sets.

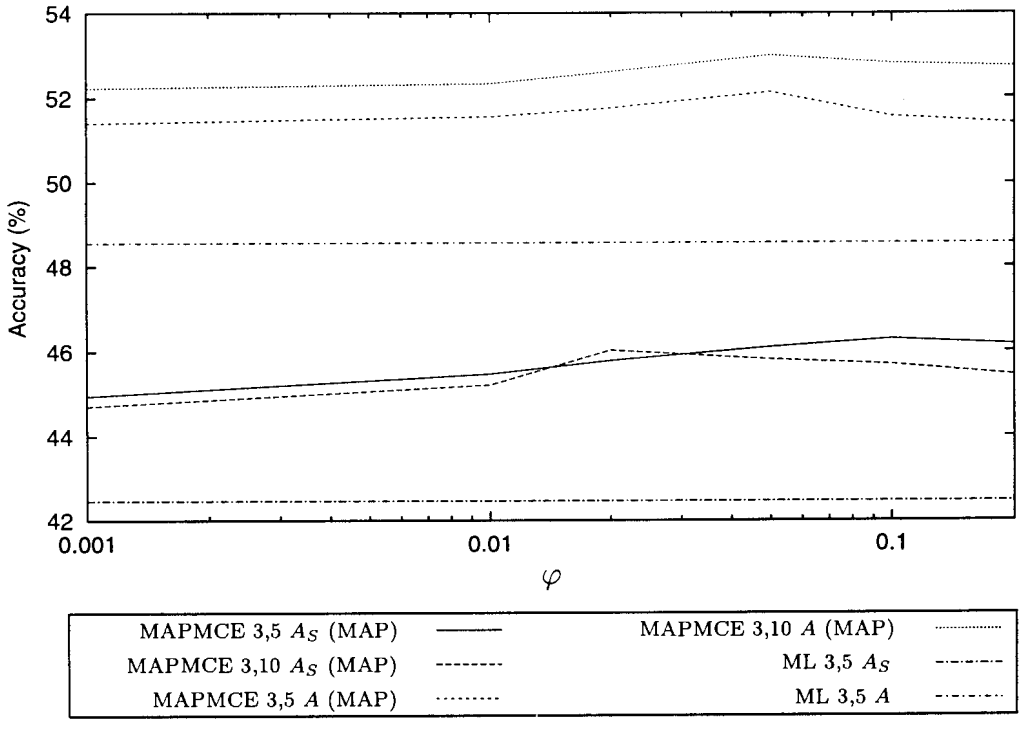


Figure 4.9: MAPMCE adaptation results for both Afrikaans training sets using the best MAP point as starting point

Comparison

Table 4.7 summarizes the best results obtained using the different algorithms for the SUNSpeech dataset. Considerable improvements are realized from the usage of the three algorithms when the small Afrikaans training set is used, with the GMAP algorithm performing best under these conditions (8.0% and 10.2% relative improvement in error rate for the 5 and 10 mixture models respectively).

Smaller, but finite, improvements are attained when the full Afrikaans training set is available. Here, the MAPMCE algorithm produces the largest improvements (6.8% and 3.1% relative improvement in error rate for the 5 and 10 mixture HMMs respectively). The MAP algorithm, although it manages to improve on the baseline performance, produces the worst results of the three algorithms for the configurations and scenarios tested. Note, however, that the MAP algorithm was integral to the optimal perfor-

Table 4.6: Summary of the results obtained using the MAPMCE algorithm

Description	Mixtures	
	5	10
MAPMCE A_S (E)	43.5%	43.3%
MAPMCE A_S (MAP)	46.3%	46.0%
MAPMCE A (MAP)	52.1%	53.0%

mance of the GMAP and MAPMCE algorithms, as the best MAP estimate was used as starting point.

Table 4.7: Summary of the best results obtained for the SUNSpeech dataset. Relative improvement in error rate over baseline is given in braces.

Description	Mixtures	
	5	10
Baseline ML A_S	42.5 (0.0%)	41.2 (0.0%)
MAP A_S $(A_S + E)$	45.4 (5.0%)	44.8 (6.1%)
GMAP A_S (MAP)	**47.1 (8.0%)**	**47.2 (10.2%)**
MAPMCE A_S (MAP)	46.3 (6.6%)	46.0 (8.2%)
Baseline ML A	48.6 (0.0%)	51.5 (0.0%)
MAP A $(A + E)$	51.1 (4.9%)	52.4 (1.9%)
GMAP A (MAP)	51.3 (5.3%)	52.5 (2.1%)
MAPMCE A (MAP)	**52.1 (6.8%)**	**53.0 (3.1%)**

4.6.2 TIMIT

The collection of large speech databases is not a trivial task (if done properly). It is not always possible to collect, segment and annotate large databases for every task or language. It is also often the case that there are imbalances in the databases. An example of one such imbalance is the fact that there is often more male speakers than

female speakers (or *vice-versa*). If there is, for example, far fewer female speakers than male speakers, then the recognizers will tend to work poorly for female speakers (as compared to performance for male speakers).

This experimental section attempts to recreate such a scenario, where relatively little female speaker training data is available, so as to test the three algorithms discussed earlier in this chapter under these conditions. Table 4.8 details the TIMIT training and testing sets used in the experiments presented in this section. Note that the default TIMIT training set has an imbalance with respect to male and female speakers, with 326 male speakers and 136 female speakers. The standard TIMIT training set is therefore used as one such example.

Table 4.8: Description of TIMIT training and testing sets used

Description	Label	Number of speakers			Sentences	Duration
		Male	Female	Total		(minutes)
Training sets:						
Full (standard)	T	326	136	462	4620	236.5
Male	T_M	326	0	326	3260	165.2
Female	T_F	0	136	136	1360	71.3
Female-small	T_{FS}	0	16	16	160	8.2
Testing set (standard)		112	56	168	1680	86.4
Male test set		112	0	112	1120	56.8
Female test set		0	56	56	560	29.7

Although there is an imbalance in the standard training set, with the female speaker accounting for only 29.5% of the training set, it is necessary to investigate the effect of an even smaller number of female speakers. It is for this purpose, that a small female training set has been created, consisting of 16 speakers (2 from each of the 8 dialect regions in TIMIT). The small female training set (T_{FS}) is therefore approximately 5% the size of the full male training set (T_M) and 12% of the full female training set (T_F). A more detailed description of the TIMIT database can be found in Section 2.4.

Two different scenarios therefore exist for the purpose of this section, namely

- The full TIMIT training set (T) is available for training purposes.
- The full male training set (T_M) and the small or reduced female training set (T_{FS}) are available for training.

The algorithms will be tested using two criteria, namely:

1. Adaptation performance - the ability of the algorithm to improve the accuracy of the test set associated with the adaptation set,
2. Training performance - the ability of the algorithm to improve the accuracy of the combined testing set ("standard" testing set in Table 4.8).

The rationale behind the above two criteria, is that although the MAP algorithm is a good adaptation technique, it will possibly not perform as well in situations where we want good performance for both adaptation and reference (prior) sets. There are numerous situations where one would prefer better combined performance. Often, for example, when a severe imbalance in the number of female and male speakers occurs and a gender independent recognizer is used, the performance for the one grouping will tend to be relatively poor (and so too the combined performance).

The combined performance, however, can be somewhat misleading. A reasonable combined performance can be produced if a system works well for one group and poorly for another. For example, a system that attains 90% correct recognition for male speakers but only 30% for female speakers would, assuming equal numbers of male and female speakers, therefore give 60% correct recognition for the combined set. The minimum performance for the two sets will also be presented, i.e. in the above example the 30% recognition rate will be reported. When summarizing results, only those algorithms

and their configurations which maximize either the female, combined and minimum testing set performance will be listed.

Table 4.9 presents the baseline ML results obtained for the TIMIT database. It is evident, as expected, that if one were to create a gender specific recognizer, that the models trained using the gender dependent subsets are best. The small female training set produces results which are worse than that obtained using the full female training set for both male and female testing sets (and consequently the combined testing set).

Table 4.9: Base system (ML) results for TIMIT

Training set	Test set accuracy			
	Male	Female	Combined	Minimum
Full training set T	57.3%	56.5%	**57.0%**	**56.5%**
Male training subset T_M	**59.1%**	43.0%	53.7%	43.0%
Female training subset T_F	40.1%	**61.0%**	47.2%	40.1%
Small female training subset T_{FS}	39.9%	56.8%	45.6%	39.9%
$T_{FS} + T_M$	59.1%	46.7%	54.9%	46.7%

The ML model resulting from the combination (pooling) of the male and the small female training subsets ($T_{FS} + T_M$) results in better combined performance than that attained by any ML model other than that of the full training dataset. The performance for the female testing set is, however, poor and it is only better than that of the ML model trained using the male training subset only.

A 3 state, 5 mixture HMM is used for all experiments using the TIMIT database in this section. The general configuration of the HMM recognizer, is as described in Section 2.1.

MAP

The usage of the MAP algorithm for both adaptation and training purposes will now be tested for both scenarios where limited training data is available for female speakers.

Figure 4.10 presents the MAP results when the small female training set is used. A pooled prior created from the pooled dataset ($T_M + T_{FS}$) is used. The performance for the female testing set peaks at $\nu = 100$, giving a peak accuracy of 57.4%. The accuracy for the male testing set improves as ν gets larger, i.e. as the prior becomes more restrictive and the final result is closer to the ML estimate obtained from the prior.

Combined performance peaks at $\nu = 2000$ giving a peaked combined performance of 56%. The solution resulting in the best combined performance would probably not be acceptable due to the disparity in results for the male and female testing sets (57.8% and 52.4% respectively). A better solution is that which maximizes the minimum performance of the two sets, this occurs for $\nu = 700$ which gives 55.1% accuracy for both male and female sets.

Figure 4.11 shows the MAP results when using the full female training set (T_F). A pooled prior ($T_M + T_F$) is used. Peak performance of 61.0% is attained for the female training set for $\nu = 50$. The performance for the male testing set, as with the previous experiment, increases as ν becomes larger.

Although the graphic does not show the point at which the male and female performance is equal or at which point maximum combined performance is produced, it will undoubtedly be very close to that at which $\nu = \infty$ or the performance of the ML model (of the pooled set).

The MAP results for the TIMIT database are summarized in Table 4.10. Using MAP for gender adaptation (improving female test set performance) when using the small training set results in a relatively small improvement in error rate (1.4% for $\nu = 100$)

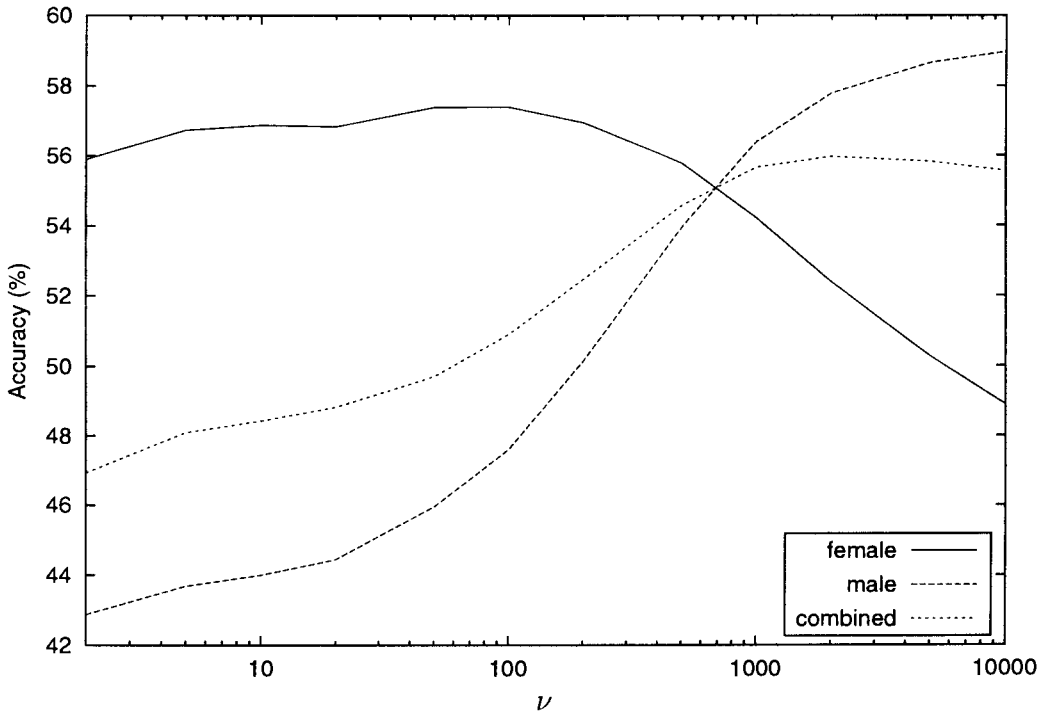


Figure 4.10: MAP adaptation results for the small female training set, using a pooled prior ($T_M + T_{FS}$)

when using the small female subset. No improvement is observed when the full female training set is used, where 61.0% accuracy is attained, equal to that realized using the ML model for the full female testing set.

A considerable improvement in minimum accuracy is attained when using reduced female training set, where a 15.8% relative reduction in error is realized (46.7% $\rightarrow$ 55.1%). Note that the best minimum accuracy (55.1%) using MAP is not much worse as that attained when using the full training set (56.5%). The MAP algorithm does not improve on the minimum accuracy attained of the ML model when using the full training set.

The results for the MAP algorithm using a prior created using the pooled dataset have been presented. The results for the MAP algorithm using a prior created from the male training set have not been included as they are similar for adaptation (female testing set accuracy) and worse in terms of the combined and minimum test set accuracies.

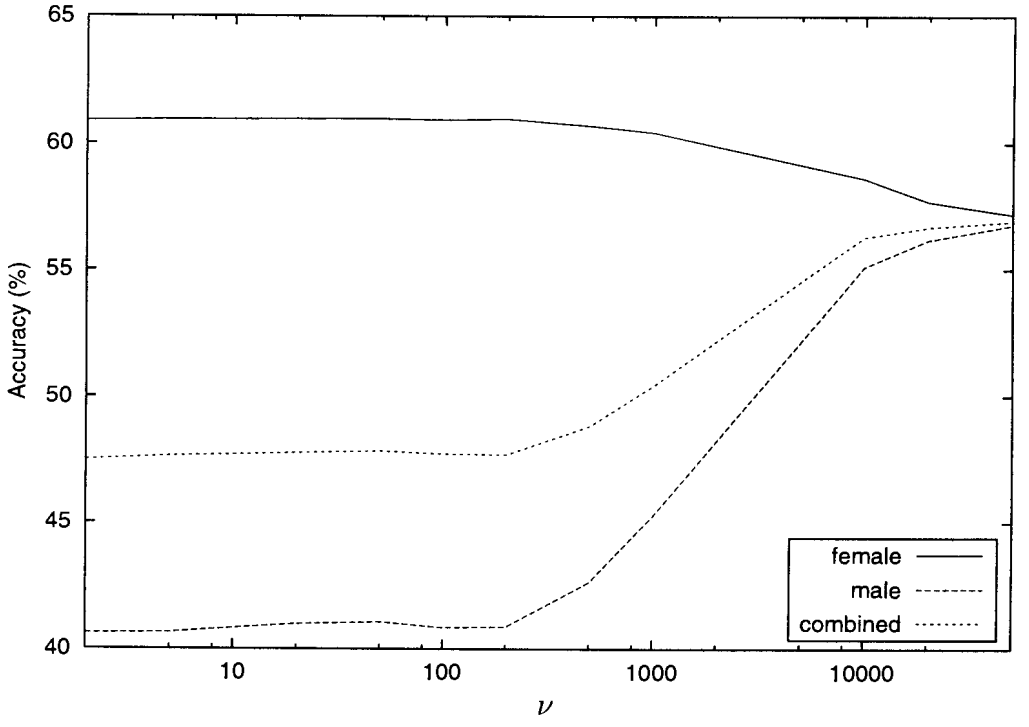


Figure 4.11: MAP adaptation results for the full female training set, using a pooled prior ($T_M + T_F$)

GMAP

The GMAP algorithm is experimentally evaluated here, for the two situations described earlier. The full male training set (T_M) is available in both situations and is used as the prior dataset throughout.

The results obtained using the GMAP algorithm for the small female training set are shown in Figure 4.12. The initial starting point used here is the ML model created from the pooled data ($T_M + T_{FS}$). The performance for the female testing set increases as φ decreases. It is therefore evident that for this scenario, the GMAP algorithm will give best performance for the female test set when $\varphi = 0$ (the ML estimate using the small female testing set).

Combined performance is maximized when $\varphi = 0.1$, producing a peak combined accu-

Table 4.10: MAP results for gender adaptation experiments using TIMIT

Algorithm	Test set accuracy (%)			
	Male	Female	Combined	Minimum
MAP T_{FS} $(T_{FS}+T_M)$, $\nu=100$	47.6	**57.4**	50.9	47.6
MAP T_{FS} $(T_{FS}+T_M)$, $\nu=700$	55.1	55.1	55.1	**55.1**
MAP T_{FS} $(T_{FS}+T_M)$, $\nu=2000$	57.8	52.4	**56.0**	52.4
MAP T_F (T_F+T_M), $\nu=50$	41.1	**61.0**	47.8	41.1
MAP T_F (T_F+T_M), $\nu=\infty$	**57.3**	56.5	**57.0**	**56.5**

racy of 56.4%, while the minimum accuracy is maximized when $\varphi = 0.02$ resulting in a 55.3% peak minimum accuracy. Although it was not an objective of this experiment, a slight improvement in male testing set performance is attained for $\varphi = 0.5$, where a 59.2% accuracy is realized.

Figure 4.13 shows the GMAP results the full female training set. The best female testing set performance for the range of φ shown occurs for $\varphi = 0.02$, where an accuracy of 60.5% is attained. The female testing accuracy for $\varphi = 0$ should, however, be the same as that of the ML model estimated using the full female training set (61.0%). The peak combined performance (57.0%) is attained when $\varphi = 1$, while the peak minimum performance (56.7%) results when $\varphi = 0.8$.

Note that the value of φ resulting in optimal minimum testing set accuracy is larger when the entire female training set is used. If there is no imbalance in the numbers of male and female speakers, then one would expect the optimal value of φ to be one (assuming that recognition accuracy would be the same for equivalent amounts of data). As more and more training data becomes available for female speakers, the contribution thereof to the posterior distribution becomes larger. The optimal value of φ therefore increases, so that the adaptation data does not dominate.

The best GMAP results using the TIMIT dataset are summarized in Table 4.11. Rea-

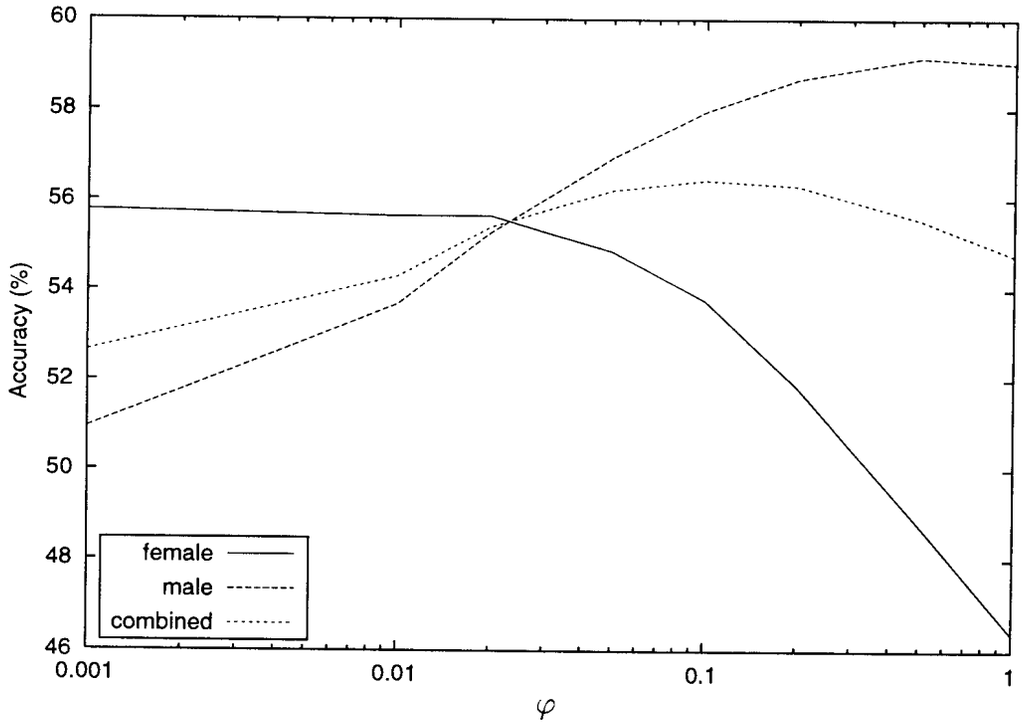


Figure 4.12: GMAP adaptation results for the small female training set, using the pooled data ($T_M + T_{FS}$) ML model as a starting point

sonable improvements in minimum and combined testing set results are realized using the GMAP algorithm with the small female training set, where a 16.1% and 3.3% relative improvement in error rates are achieved respectively.

The GMAP algorithm, however, failed to improve on the female testing set performance attained using the ML trained model using the corresponding female training set, for both scenarios where the small and full female training sets are used.

MAPMCE

The MAPMCE algorithm presented in Section 4.5 is tested using the TIMIT dataset given the conditions described earlier. The male speaker training set is used as the prior dataset for all experiments conducted in this section.

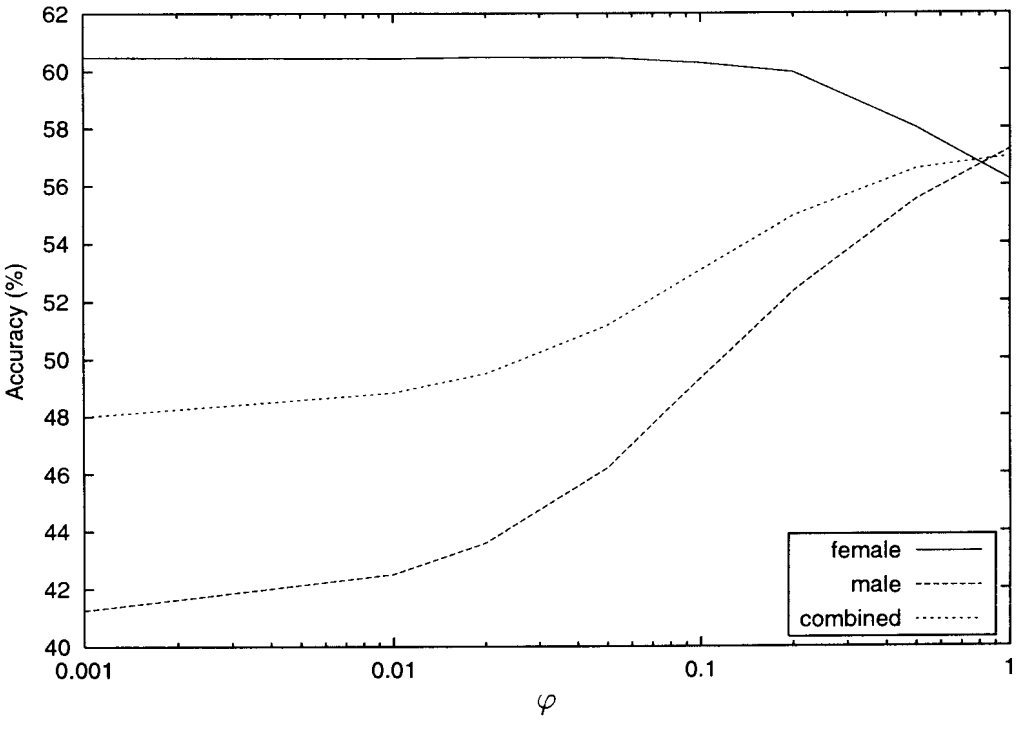


Figure 4.13: GMAP adaptation results for the full female training set, using a pooled data ($T_M + T_F$) ML model as starting point

Figure 4.14 presents the MAPMCE results for the small female training set using the pooled ML model as a starting point. Note that here the female testing set performance does not improve to the extent that it becomes better than that of the male testing set. The peak minimum and female testing set performances are therefore obtained for the same value of $\varphi = 0.1$, which produces a peak accuracy of 56.3% for both criteria. The best combined performance of 62.2% is attained when $\varphi = 0.5$, though this is only slightly better than that attained at the peak minimum accuracy point, which results in a combined performance of 62.1%.

Figure 4.15 shows the MAPMCE results for the small female training set, but with the best MAP point used as starting point. Here, the female testing set performance is considerably better, with a peak performance of 60.3% being attained for $\varphi = 0.2$. The accuracies attained for the male testing set are, however, somewhat worse with a peak male testing set accuracy of 63.0% resulting for $\varphi = 1.0$. The best combined

Table 4.11: GMAP results for gender adaptation experiments using TIMIT

Algorithm	Test set accuracy (%)			
	Male	Female	Combined	Minimum
GMAP T_{FS} $(T_{FS}+T_M)$, $\varphi = 0.001$	50.5	**55.8**	52.6	50.5
GMAP T_{FS} $(T_{FS}+T_M)$, $\varphi = 0.02$	55.3	55.6	55.4	**55.3**
GMAP T_{FS} $(T_{FS}+T_M)$, $\varphi = 0.1$	58.0	53.8	**56.4**	53.8
GMAP T_{FS} (MAP), $\varphi = 0.01$	46.2	**57.0**	50.0	46.2
GMAP T_{FS} (MAP), $\varphi = 0.08$	55.0	55.1	55.1	**55.0**
GMAP T_{FS} (MAP), $\varphi = 0.2$	57.2	52.0	**55.4**	52.0
GMAP T_F (T_F+T_M), $\varphi = 0.02$	43.6	**60.5**	49.5	43.6
GMAP T_F (T_F+T_M), $\varphi = 0.8$	56.7	56.7	56.7	**56.7**
GMAP T_F (T_F+T_M), $\varphi = 1.0$	**57.2**	56.2	**57.0**	56.2
GMAP T_F (MAP), $\varphi = 0.01$	41.2	**60.9**	47.8	41.2
GMAP T_F (MAP), $\varphi = 0.7$	57.0	57.0	57.0	**57.0**
GMAP T_F (MAP), $\varphi = 1.0$	**57.7**	55.9	**57.1**	55.9

accuracy is 61.4% for $\varphi = 1.0$, which is worse than that produced by the MAPMCE algorithm using the ML model trained using the pooled dataset as starting point.

The peak minimum test set accuracy is 60.3% ($\varphi = 0.2$), which is considerably better than that attained (56.3%) using the pool dataset ML model as starting point. Although the combined testing set performance is better using the previous variant of the MAPMCE algorithm, one would rather choose to use the later version due to the considerably improved minimum and female testing set performance.

Table 4.12 summarizes the MAPMCE results using both the small and full female training sets. Large improvements in the minimum, combined and female testing set performances are attained for both the small and full female training sets. Using the MAP point estimate as starting point produced the best female and minimum testing set performances. Maximum combined testing set performance was, however, attained

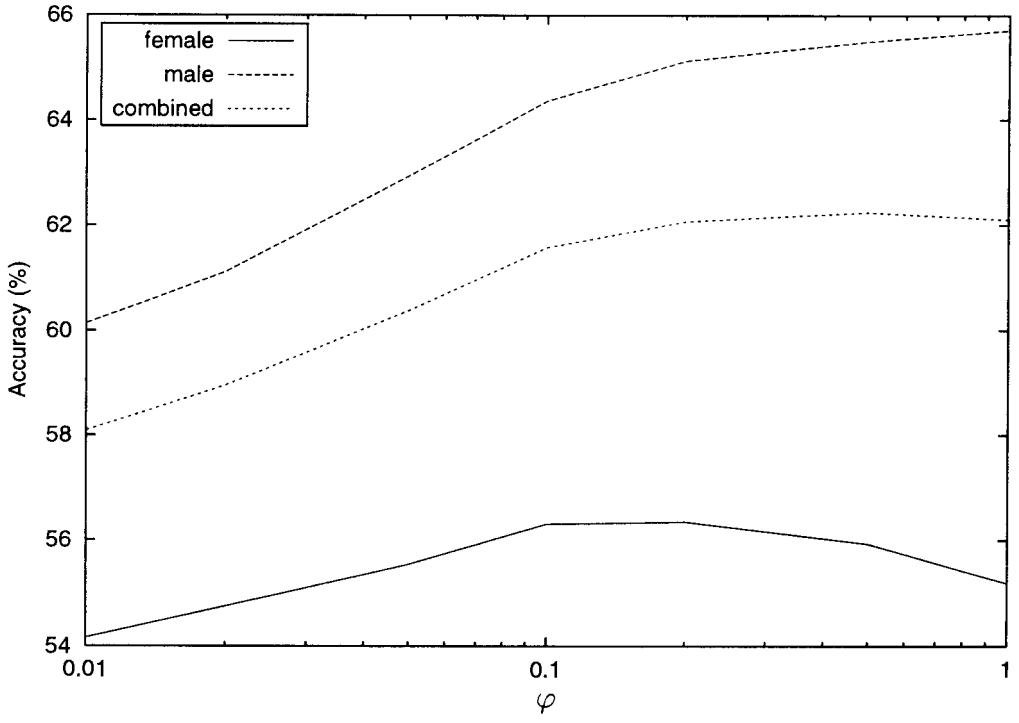


Figure 4.14: MAPMCE adaptation results for the small female training set, using the full male training set as prior (T_M) with the pooled ML model as starting point

when the ML model trained using the pooled data was used as starting point.

Comparison

Table 4.13 summarizes the best adaptation results (female testing set accuracy) obtained using the different algorithms for the TIMIT dataset. The MAP and GMAP algorithms did not improve female testing set performance when the full female training set was used. MAP improved the female test set accuracy to a limited degree when the small female training set was available. The MAPMCE algorithm performed best, with relative improvements in error rate of 8.1% and 13.1% resulting for the small T_{FS} and full T_F female training sets respectively.

Table 4.14 summarizes the best training results (minimum testing set accuracy) obtained using the different algorithms for the TIMIT dataset. The MAP and GMAP

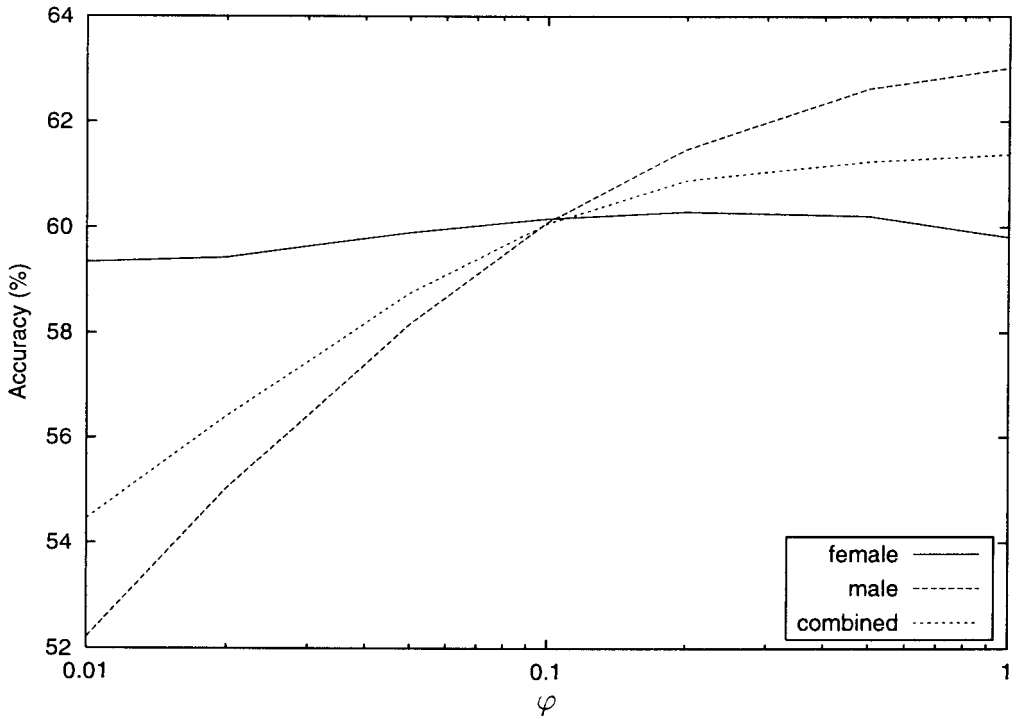


Figure 4.15: MAPMCE adaptation results for the small female training set, using the full male training set as prior (T_M) with the best MAP estimate as starting point

algorithm produced reasonable improvements in error rate when the small female training set was used (15.8% and 16.1% respectively). Little or no improvement was attained using these two algorithms when the full female training set was used. MAPMCE, once again produced marked improvements in the minimum testing set accuracy, with 25.5% and 14.5% relative improvement in error rate for the small and full female training sets respectively.

4.6.3 TIDIGITS

In this section, the three algorithms (MAP, GMAP and MAPMCE) will be compared within the framework of a continuous digit recognition task. As in the previous section, a situation is created where the number of female speakers is limited. The adaptation algorithms are therefore required to improve female test set performance, as well as for

Table 4.12: MAPMCE results for gender adaptation experiments using TIMIT

Description	Test set accuracy (%)			
	Male	Female	Combined	Minimum
MAPMCE T_{FS} $(T_{FS}+T_M)$, $\varphi = 0.2$	65.1	**56.3**	62.1	**56.3**
MAPMCE T_{FS} $(T_{FS}+T_M)$, $\varphi = 0.5$	65.5	55.9	**62.2**	55.9
MAPMCE T_{FS} $(T_{FS}+T_M)$, $\varphi = 1.0$	**65.7**	55.2	62.1	55.2
MAPMCE T_{FS} (MAP) $(\varphi = 0.2)$	61.5	**60.3**	60.9	**60.3**
MAPMCE T_{FS} (MAP) $(\varphi = 1.0)$	**63.0**	59.8	**61.4**	59.8
MAPMCE T_F (MAP) $(\varphi = 0.2)$	60.0	**66.1**	61.8	60.0
MAPMCE T_F (MAP) $(\varphi = 1.0)$	**62.8**	65.1	**63.4**	**62.8**

the combined and minimum test set performance.

Table 2.6 presents the training and testing sets used in the experiments conducted in this section. The standard TIDIGITS training set comprises 77 digit sequences from each of the 57 female speakers in the dataset and 55 male speakers. There is therefore not an imbalance in the numbers of male and female speakers. Two subsets of the female training set are therefore created, so as to simulate the situation where relatively little data from female speakers is available.

The first subset (T_{WS}) consists of five female speakers selected at random from the complete female training set and all of the associated 77 digit sequences. The total duration of the resultant training set (T_{WS}) is 10.1 minutes long, which is approximately 7.7% the size of the full female training set. The second, smaller female training set (T_{WVS}) is a subset of the first, where the digit sequences of each of the speakers has been reduced to one each of the 1,2,3,4,5 and 7 digit sequences per speaker (randomly selected). The number of speakers is also reduced to 4, resulting in a female training set which is 55 seconds in duration and approximately one tenth the size of the first female training subset.

Table 4.13: Summary of the best adaptation results obtained for the TIMIT dataset. Relative improvement in error rate is given in braces.

Description	Female test set accuracy (%)
Baseline ML T_{FS}	56.8 (0.0%)
MAP T_{FS} $(T_{FS} + T_M)$, $\nu = 100$	57.4 (1.4%)
GMAP T_{FS} $(T_{FS} + T_M)$, $\varphi = 0$	56.8 (0.0%)
MAPMCE T_{FS} (MAP), $\varphi = 0.2$	**60.3 (8.1%)**
Baseline ML T_F	61.0 (0.0%)
MAP T_F $(T_F + T_M)$, $\nu = 50$	61.0 (0.0%)
GMAP T_F $(T_F + T_M)$, $\varphi = 0$	61.0 (0.0%)
MAPMCE T_F (MAP), $\varphi = 0.2$	**66.1 (13.1%)**

The second subset (T_{WVS}) is extremely small, but it will be shown that reasonable performance can be realized even for such a small training set. Note that although the female training sets are small the full male training set is assumed to be available for training purposes.

As in the last section, the algorithms will be compared by their ability to adapt models (female testing set performance) and performance of gender independent training (combined and minimum test set accuracy).

Table 4.16 details the performance of an 8 state, 5 mixture ML trained HMM trained using the datasets described above. The results using the full training sets are presented as a reference for the experimental results using the reduced datasets.

Reasonable accuracies are realized when only the small female training set (T_{WS}) is available. Peak female testing set accuracy of 95.6% is attained when using a model trained using only the small female training set. Pooling the small female training set and the full male training set ($T_M + T_{WS}$) decreases the performance for the female testing set slightly, but improves male, combined and minimum testing set performances markedly.

Table 4.14: Summary of the best minimum test set accuracy obtained for the TIMIT dataset. Relative improvement in error rate is given in braces.

Description	Minimum test set accuracy (%)
Baseline ML $T_{FS} + T_M$	46.7 (0.0%)
MAP T_{FS} $(T_{FS} + T_M)$, $\nu = 700$	55.1 (15.8%)
GMAP T_{FS} $(T_{FS} + T_M)$, $\varphi = 0.02$	55.3 (16.1%)
MAPMCE T_{FS} (MAP), $\varphi = 0.2$	**60.3 (25.5%)**
Baseline ML T	56.5 (0.0%)
MAP T_F $(T_F + T_M)$, $\nu = 50$	56.5 (0.0%)
GMAP T_F $(T_F + T_M)$, $\varphi = 0.8$	56.7 (0.5%)
MAPMCE T_F (MAP), $\varphi = 1.0$	**62.8 (14.5%)**

Using the very small female training set (T_{WVS}) results in poor results (25.2% accuracy for the female testing set). Pooling with the male training set, once again greatly improves results. It is, however, interesting to note that the ML trained model using only the male training set results in better results.

When computing the relative improvement obtained using the algorithms, the best results obtained when using only the available training data will be used. Table 4.17 shows the best ML results that can be achieved for the two scenarios, where the amount of female training data is limited. Note that, as mentioned, the best results obtained when the very small female training set is available is that of the ML model trained using only the male training set and the associated results are therefore used.

As discussed, the combined performance is not a good measure of the suitability of a particular model or algorithm. The ML model trained using only the male (or female) training set is a good example of this problem, a combined performance of 90.6% seems reasonable, but there is a large difference between female and male testing set performance (97.4% vs. 84.0%), which is not desired.

Table 4.15: Training and testing sets used with the TIDIGIT database

Description	Label	Number of speakers			Digit sequences	Duration (minutes)
		Male	Female	Total		
Training sets:						
Full (standard)	T	55	57	112	8623	253.4
Man	T_M	55	0	55	4235	121.9
Woman	T_W	0	57	57	4389	131.5
Woman-small	T_{WS}	0	5	5	385	10.1
Woman-very-small	T_{WVS}	0	4	4	28	55 s
Testing set		55	57	113	8623	254.4

MAP

The standard MAP algorithm presented in Section 4.3 is now tested within connected digit recognition task using the TIDIGITS database. As with the previous databases used to test the MAP algorithm, a pooled prior was found to perform better than that created from only the reference (male) training set. The prior used in all experiments presented in this section is therefore created using the pooled dataset containing the male training set and the relevant female training set.

Figure 4.16 shows the MAP adaptation results for the scenario where the small female training set (T_{WS}) is available. The pooled data set ($T_{WS} + T_M$) is used to create the prior. Peak female testing set performance of 97.4% is attained for $\nu = 200$, which also results in a peak combined performance of 97.2% The peak minimum accuracy of 97.2% occurs at $\nu = 300$, relatively close to the combined performance peak.

Figure 4.17 presents the MAP adaptation results for the scenario where the smallest female training set (T_{WVS}) is available. The amount of female adaptation data is extremely small, and as a result the female testing set performance cannot be improved to the extent that it is better than that of the male testing set (for the specific config-

Table 4.16: Base system results for an 8 state, 5 mixture HMM using TIDIGITS

Training set	Test set accuracy (%)			
	Male	Female	Combined	Minimum
Full training set (T)	**97.5**	98.4	**97.9**	**97.5**
Man, training set (T_M)	97.4	84.0	90.6	84.0
Woman, training set (T_W)	90.3	**98.8**	94.6	90.3
Woman, small training set (T_{WS})	78.0	**95.6**	87.0	78.0
$T_M + T_{WS}$	**97.6**	94.4	**95.9**	**94.4**
Woman, very small set (T_{WVS})	15.6	25.2	20.5	15.6
$T_M + T_{WVS}$	**91.3**	**80.1**	**85.6**	**80.1**

Table 4.17: Best base system results, given the available data

Available data	Test set accuracy (%)			
	Male	Female	Combined	Minimum
T_M, T_{WS}	97.6	95.6	95.9	94.4
T_M, T_{WVS}	97.4	84.0	90.6	84.0

uration). Peak female, combined and minimum testing set accuracies of 84.4%, 87.2% and 84.4% respectively are attained for $\nu = 200$. The female and minimum testing set performances are slightly better than that of the ML model trained using the male training set only (a relative improvement in error rate of 2.5%). The combined test set accuracy is, however, considerably worse than that obtained using the best ML model (-36.2%).

Table 4.18 summarizes the results obtained using the MAP adaptation algorithm for the two training datasets. Reasonable improvements in error rates are attained when the larger female training set (T_{WS}) is available. However, the MAP algorithm fails to significantly improve the minimum and female testing set accuracy when the smallest female training (T_{WVS}) set is used. Importantly, a decrease in combined performance

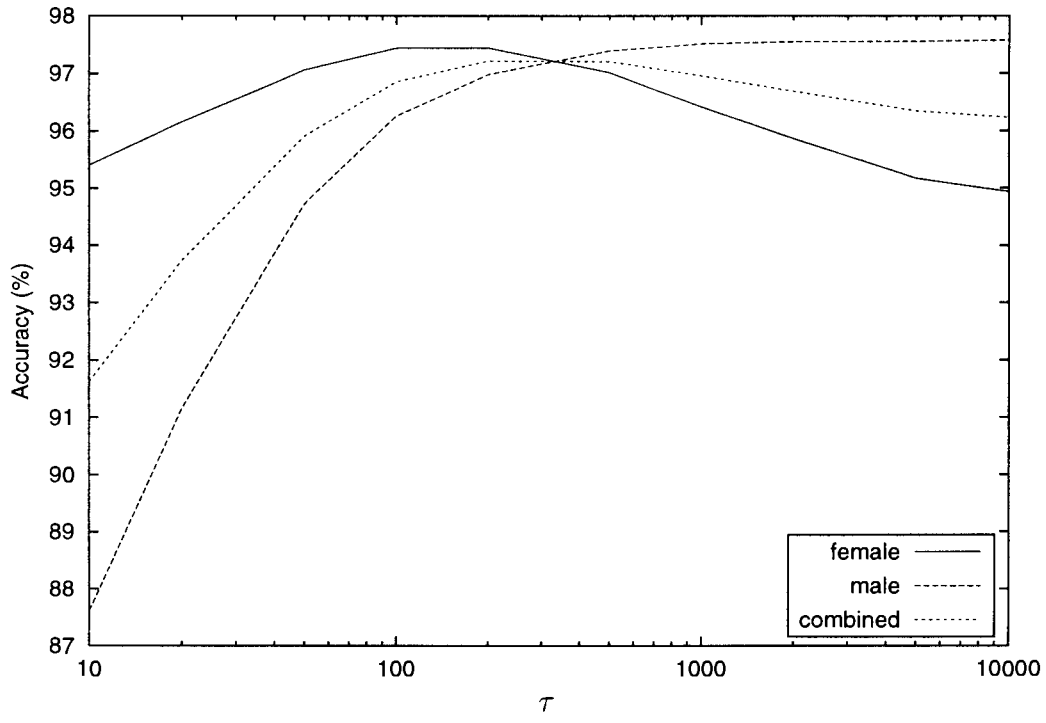


Figure 4.16: MAP adaptation results for an 8 state, 5 mixture HMM using the small female training set (T_{WS}) as the adaptation set and the pooled training set ($T_{WS} + T_M$) to create the prior.

results when using the MAP algorithm in the extreme situation, where the smallest female training set is used.

GMAP

The GMAP algorithm presented in Section 4.4 is tested using the TIDIGIT dataset given the conditions described earlier. The male speaker training set is used as the prior dataset for all experiments conducted in this section.

Figure 4.18 details the GMAP adaptation results for an 8 state, 5 mixture HMM when the small female training set (T_{WS}) is available. The algorithm attains peak performance of 97.6% for the female testing set when $\varphi = 0.05$, which is a 45.5% relative improvement in error rate. A slight, though insignificant improvement in accuracy for

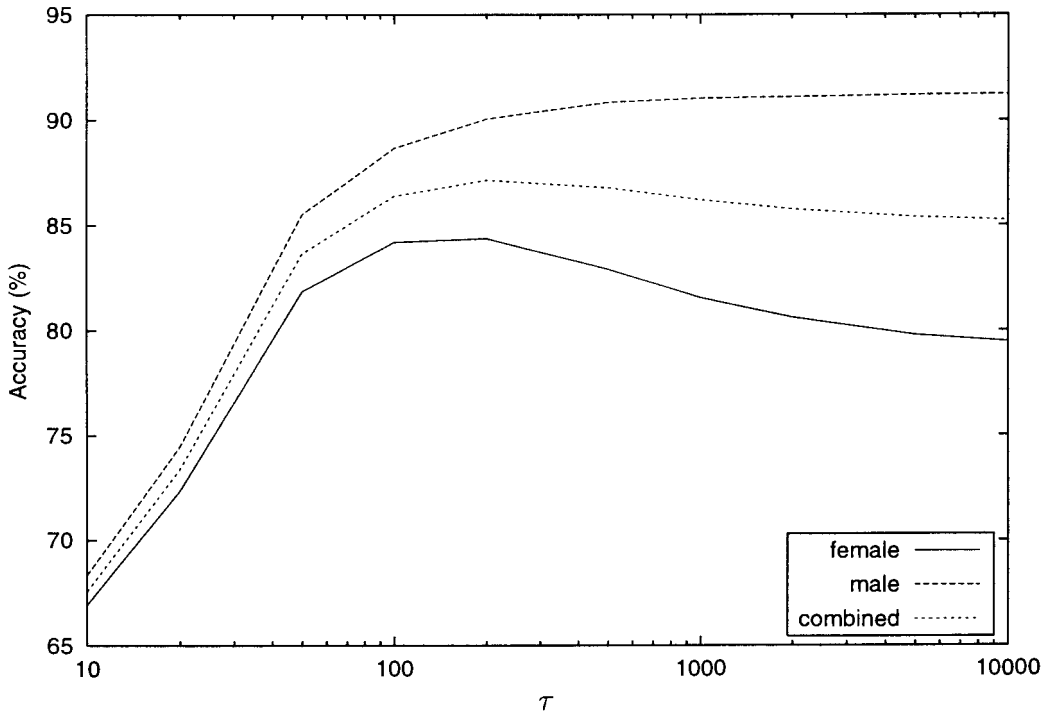


Figure 4.17: MAP adaptation results when the smallest female training set (T_{WVS}) is available

the male testing set can be attained if we set $\varphi = 0.5$. Peak combined and minimum testing set performances both occur when $\varphi = 0.2$, where a 97.5% accuracy is realized (for both). This equates to a 39% and 55.4% relative improvement in error rate for the combined and minimum testing set accuracies respectively.

The GMAP results for scenario where the very small female training set is available are presented in Figure 4.19. Here, as with MAP, the female testing set performance can not be improved to the degree that it is higher than that for the male testing set (for the range of φ presented). One would, however, expect the male training set performance to drop below that of the female testing set as $\varphi \to 0$, which is the ML estimate using only the female training data. A peak female testing set accuracy of 88.8% results for $\varphi = 0.01$, which is a 30.0% relative improvement in error rate (compared to the male training set (T_M) ML model).

Consequently, the minimum testing set performance peaks at the same point and accu-

Table 4.18: MAP results for an 8 state, 5 mixture HMM for TIDIGITS

Description	Test set accuracy (%)			
	Male	Female	Combined	Minimum
MAP 8,5 $T_{WS}(T_{WS}+T_M),\nu=200$	97.0	**97.4**	**97.2**	97.0
MAP 8,5 $T_{WS}(T_{WS}+T_M),\nu=300$	97.2	97.2	97.2	**97.2**
MAP 8,5 $T_{WS}(T_{WS}+T_M),\nu=10^5$	**97.6**	94.9	96.2	94.9
MAP 8,5 $T_{WVS}(T_{WVS}+T_M),\nu=200$	90.0	**84.4**	**87.2**	**84.4**
MAP 8,5 $T_{WVS}(T_{WVS}+T_M),\nu=10^5$	**91.2**	79.5	85.2	79.5

racy as that of the female testing set (88.8% at $\varphi = 0.01$), which is also a 30.0% in error rate. The combined performance is a maximum at either $\varphi = 0.02$ or $\varphi = 0.01$ where a combined accuracy of 89.9% results (a relative increase in error rate of 7.4%). Here, the combined error rate is still worse than that attained using the ML model trained using the male dataset, though as discussed, this is not a good measure of model or algorithm performance.

Table 4.19 summarizes the results obtained using the GMAP algorithms for the various scenarios created using the TIDIGIT dataset. The results for the GMAP algorithm using the ML model of the pooled dataset ($T_{WVS}+T_M$) as a starting point are also presented (for the small female dataset). It is noticeable that here, the usage of the best MAP point estimate results in a small, but significant improvement in error rates (for the three criteria used to evaluate the algorithms).

MAPMCE

The MAPMCE algorithm presented in Section 4.5 is tested using the TIDIGIT dataset given the conditions described earlier in this section. The male speaker training set is used as the prior dataset for all experiments conducted in this section.

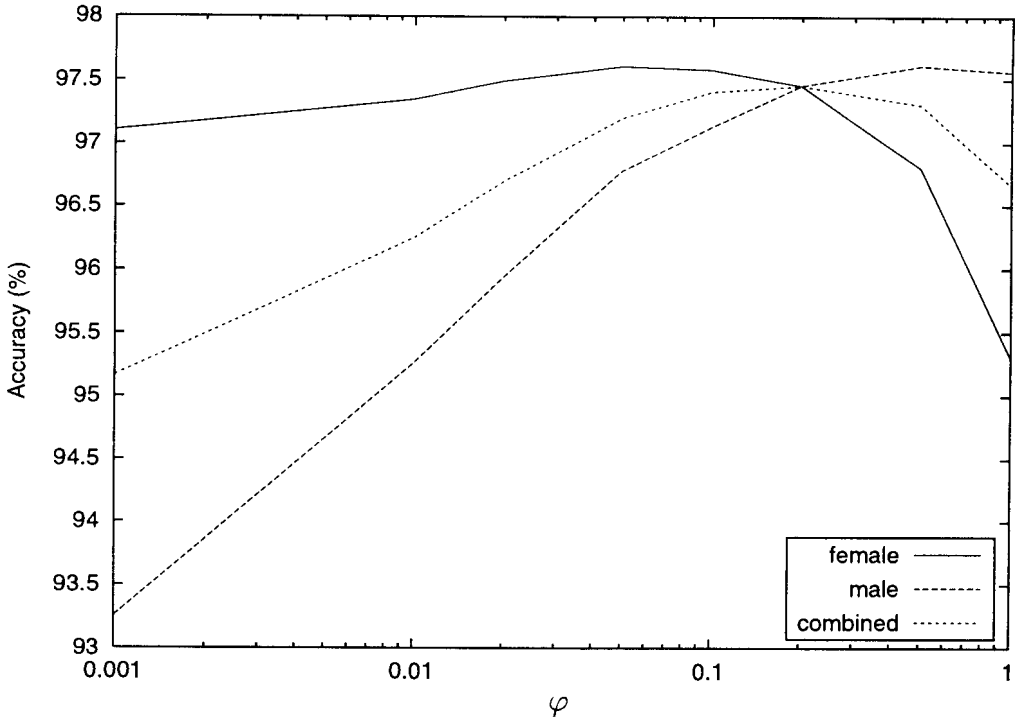


Figure 4.18: GMAP adaptation results for an 8 state, 5 mixture HMM using the small female training set (T_{WS}) as the adaptation set and the male training set (T_M) as prior. The best MAP point is used as starting point.

Figure 4.20 presents the MAPMCE results for an 8 state, 5 mixture HMM when the small female training set is available. The male training set is used as the prior dataset and the best MAP estimate is used as a starting point. It is interesting to note that the female testing set accuracy is above that of the male testing set for the range of φ presented. The male, female, combined and minimum test set accuracies peak at $\varphi = 0.5$, where their respective accuracies are 98.1%, 98.5%, 98.3% and 98.1%. The associated relative improvement in error rate is therefore 65.9% for the female testing set, 58.5% for the combined testing set performance and 66.1% for the minimum testing set accuracy.

The female testing set results are, however, considerably worse than that of the male testing set when using the MAPMCE algorithm for the scenario where the very small female training set is available, as shown in Figure 4.21. The peak female, combined and minimum testing set accuracies all occur at $\varphi = 0.05$. A peak female testing set

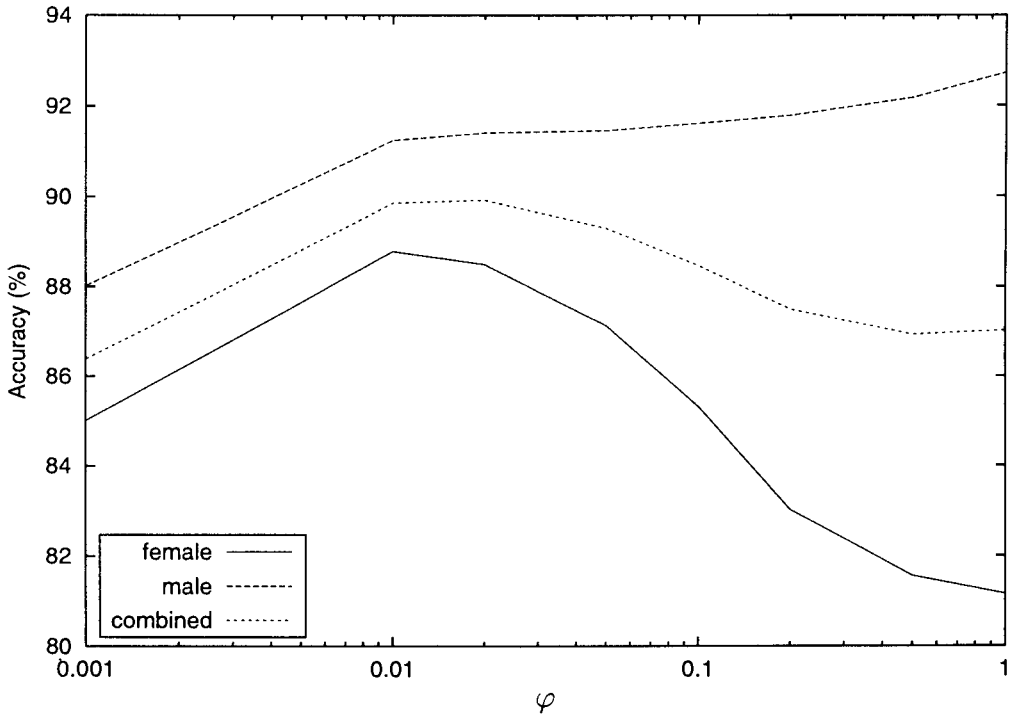


Figure 4.19: GMAP adaptation results for an 8 state, 5 mixture HMM using the very small female training set (T_{WVS}) as the adaptation set and the male training set (T_M) as prior. The ML estimate trained using the pooled training set is used as starting point.

accuracy of 93.0% is attained, which equates to a 56.3% relative improvement in error rate. Large improvements are also realized for the combined and minimum testing set performance measures, where a 95.1% combined testing set accuracy (47.9% relative improvement) and 93.0% minimum testing set accuracy (56.3% relative improvement) are attained.

Table 4.20 summarizes the results obtained using the MAPMCE algorithm under the two scenarios created using the TIDIGITS dataset. The results for the MAPMCE algorithm using the ML model of the pooled dataset ($T_{WS} + T_M$) as starting point are also presented (for the small female training set). Once again, as with the GMAP algorithm, using the best MAP point as the starting point has proved to be a better choice (compared to using the ML estimate of the pooled dataset). The results for the MAPMCE algorithm for the smallest female training set using the best MAP point as starting point have not been included as they are very similar to that obtained using

Table 4.19: GMAP results for an 8 state, 5 mixture HMM for TIDIGITS

Description	Test set accuracy (%)			
	Male	Female	Combined	Minimum
GMAP T_{WS} (MAP), $\varphi = 0.05$	96.8	**97.6**	97.2	96.8
GMAP T_{WS} (MAP), $\varphi = 0.2$	97.5	97.5	**97.5**	**97.5**
GMAP T_{WS} (MAP), $\varphi = 0.5$	**97.6**	96.8	97.3	96.8
GMAP T_{WS} ($T_{WS} + T_M$), $\varphi = 0.05$	97.4	**97.4**	**97.4**	**97.4**
GMAP T_{WS} ($T_{WS} + T_M$), $\varphi = 1.0$	**97.6**	95.4	95.9	95.4
GMAP T_{WVS} ($T_{WVS} + T_M$), $\varphi = 0.01$	91.2	**88.8**	**89.9**	**88.8**
GMAP T_{WVS} ($T_{WVS} + T_M$), $\varphi = 1.0$	**92.7**	81.2	87.0	81.2

the ML model of the pooled dataset.

Table 4.20: MAPMCE results for an 8 state, 5 mixture HMM for TIDIGITS

Description	Test set accuracy (%)			
	Male	Female	Combined	Minimum
MAPMCE $T_{WS}(T_{WS} + T_M)$, $\varphi = 0.05$	98.7	**97.7**	**98.2**	**97.7**
MAPMCE $T_{WS}(T_{WS} + T_M)$, $\varphi = 0.5$	**98.7**	97.2	97.9	97.2
MAPMCE T_{WS}(MAP), $\varphi = 0.5$	**98.1**	**98.5**	**98.3**	**98.1**
MAPMCE $T_{WVS}(T_{WVS} + T_M)$, $\varphi = 0.05$	97.9	**93.0**	**95.1**	**93.0**
MAPMCE $T_{WVS}(T_{WVS} + T_M)$, $\varphi = 1.0$	**97.9**	91.1	93.8	91.1

Comparison

Table 4.21 summarizes the female testing set results obtained for the TIDIGITS dataset using the three algorithms which have been evaluated in this section. The best minimum testing set accuracies are summarized in Table 4.22. Looking at the normal results, it is evident that the MAPMCE algorithm performs best, with the GMAP algorithm performing well.

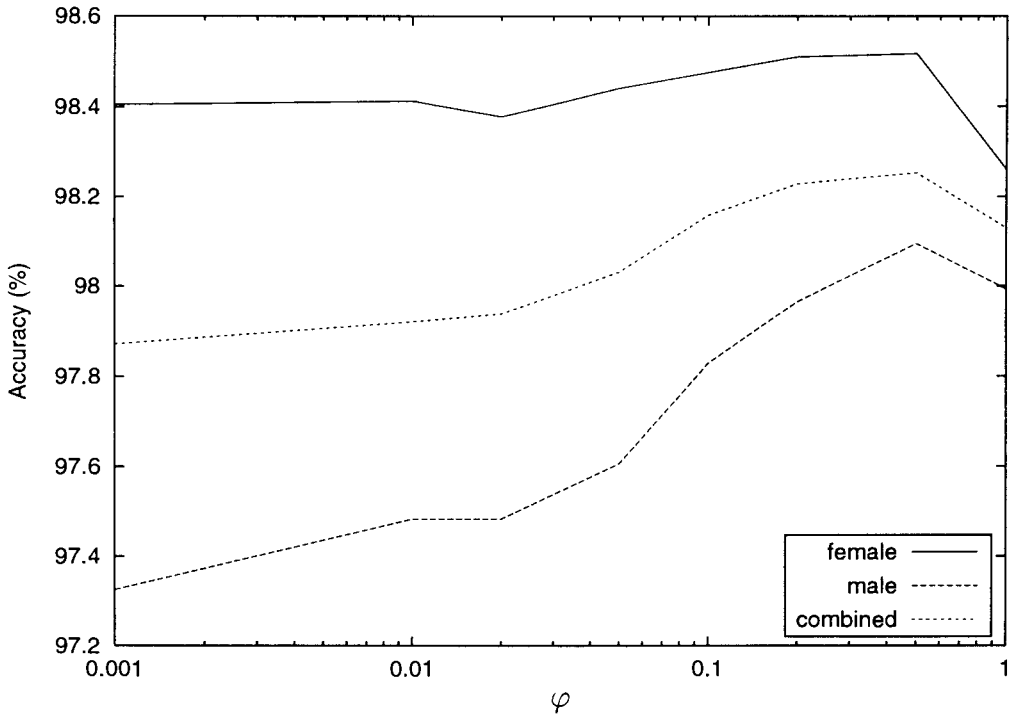


Figure 4.20: MAPMCE adaptation results for an 8 state, 5 mixture HMM using the small female training set (T_{WS}) as the adaptation set and the male training set (T_M) as prior. The best MAP point is used as starting point.

The MAP algorithm, although it does not perform as well as the other algorithms, does manage reasonable improvements in testing set performances for the larger, small female training set. It does not, however, significantly improve results when the very small female training set is used. The GMAP and MAPMCE algorithms produce far better results under these extreme conditions.

Duration modeling Duration modeling is a technique, which is often used to improve recognition accuracy in continuous digit recognition applications. It is therefore important to determine the effect of duration modeling on the performance of the algorithms tested. Note that the MAPMCE (and GMAP) algorithm can be used to estimate the duration modeling parameters. So as to ensure a fair comparison, this potential improvement has not been used and the duration parameters are therefore estimated with the ML algorithm using the relevant model and available training data.

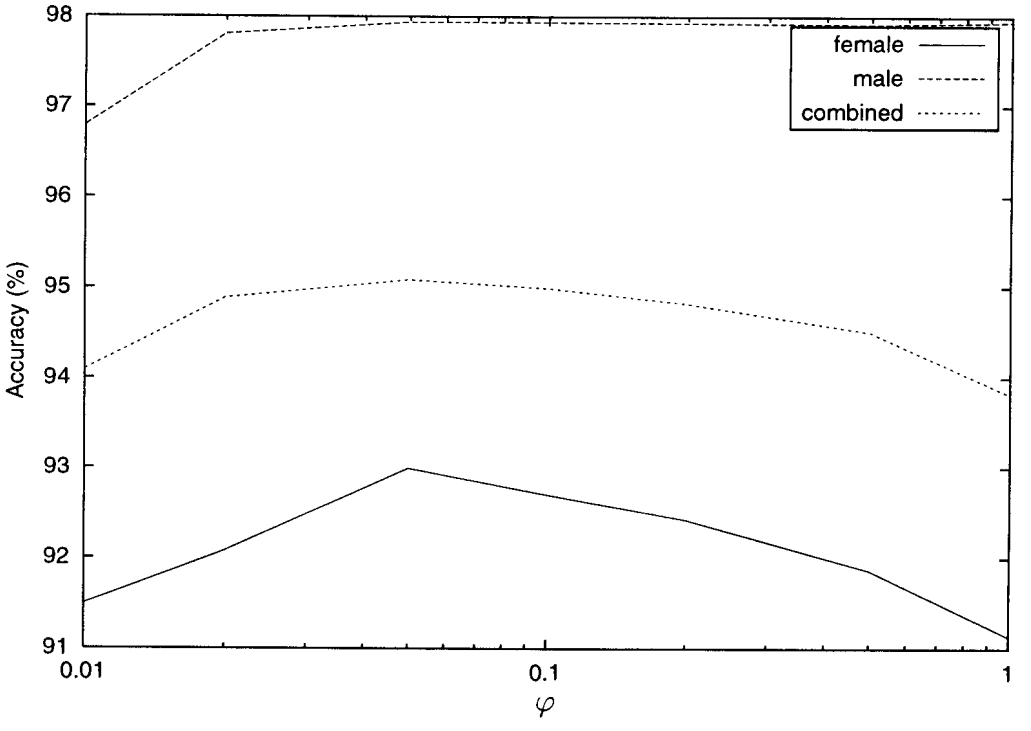


Figure 4.21: MAPMCE adaptation results for an 8 state, 5 mixture HMM using the very small female training set (T_{WVS}) as the adaptation set and the male training set (T_M) as prior. The best MAP point is used as starting point.

It is evident, from tables 4.21 and 4.22 that MAPMCE does not perform as well when used with duration modeling, and even results in a decrease in accuracy for the minimum test set accuracy measure when used with the small female training set (compared to the baseline system with duration modeling).

The MAP algorithm works better, both in absolute and relative terms, when using duration modeling. The GMAP algorithm, however, works best of all when used with duration modeling, resulting in recognition performances which are considerably better than that realized by the MAPMCE and MAP algorithms.

The MCE (MAPMCE) algorithm, due to its discriminative nature, incorporates duration information into the HMM without us explicitly modeling it. This is as a result of the MCE algorithm trying to reduce insertions and deletions. Much of the improvements realized when using the MCE algorithm therefore overlap with those attained

Table 4.21: Comparison of best results for the female test set, with and without duration modeling. The relative improvement in error rate compared to the baseline ML accuracy is given in brackets.

Description	Female test set accuracy (%)	
	Normal	Duration
ML $T_M + T_{WS}$	95.6 (0.0%)	97.6 (0.0%)
MAP T_{WS} ($T_{WS} + T_M$), $\nu = 200$	97.4 (40.9%)	**98.5 (37.5%)**
GMAP T_{WS} (MAP), $\varphi = 0.05$	97.6 (45.5%)	**98.5 (37.5%)**
MAPMCE T_{WS} (MAP), $\varphi = 0.5$	**98.5 (65.9%)**	**98.5 (37.5%)**
ML T_M	84.0 (0.0%)	91.8 (0.0%)
MAP T_{WVS} ($T_{WVS} + T_M$), $\nu = 200$	84.4 (2.5%)	94.6 (34.1%)
GMAP T_{WVS} ($T_{WVS} + T_M$), $\varphi = 0.01$	88.8 (30.0%)	**96.3 (54.9%)**
MAPMCE T_{WVS} ($T_{WVS} + T_M$), $\varphi = 0.05$	**93.0 (56.3%)**	94.4 (31.7%)

when using duration modeling. It is for this reason that the MAPMCE algorithm does not work as well when duration modeling is used.

4.6.4 Summary of results

In Section 4.6.1 the three algorithms, MAP, GMAP and MAPMCE were used within a language adaptation framework. The algorithms were used to adapt English seed data or models to create a language specific recognizer for Afrikaans. Either the full Afrikaans training set or a reduced subset thereof was used as the adaptation set.

The MAPMCE algorithm performed best when the full Afrikaans dataset was used, resulting in relative improvements in error rate of up to 6.8% and 3.1% for a 5 and 10 mixture HMM respectively (3 states). The GMAP algorithm resulted in reasonable performance increases, with a 5.3% and 2.1% relative improvement in error rate being attained for a 5 and 10 mixture HMM respectively. The standard MAP algorithm, resulted in similar, but slightly worse, improvements as that produced by the GMAP

Table 4.22: Comparison of best results for the minimum accuracy criterion, with and without duration modeling. The improvement in error rate due to the usage of duration modeling is also given.

Description	Minimum accuracy (%)	
	Normal	Duration
ML $T_M + T_{WS}$	94.4 (0.0%)	97.6 (0.0%)
MAP T_{WS} ($T_{WS} + T_M$), $\nu = 200$	97.0 (46.4%)	97.9 (12.5%)
GMAP T_{WS} (MAP), $\varphi = 0.2$	97.5 (55.3%)	**98.4 (33.3%)**
MAPMCE T_{WS} (MAP), $\varphi = 0.5$	**98.1 (66.0%)**	97.5 (-4.2%)
ML T_M	84.0 (0.0%)	91.8 (0.0%)
MAP T_{WVS} ($T_{WVS} + T_M$), $\nu = 200$	84.4 (2.5%)	94.6 (34.1%)
GMAP T_{WVS} ($T_{WVS} + T_M$), $\varphi = 0.01$	88.8 (30.0%)	**96.3 (54.9%)**
MAPMCE T_{WVS} ($T_{WVS} + T_M$), $\varphi = 0.05$	**93.0 (56.3%)**	94.4 (31.7%)

algorithm.

When the small Afrikaans training set was used, the MAPMCE algorithm did not perform as well as the GMAP algorithm, which produced relative improvements in error rate of 8.0% and 10.2% for a 5 and 10 mixture HMMs respectively. The MAP algorithm resulted in relatively poor improvements in accuracy (compared to the GMAP algorithm).

In Section 4.6.2 the three algorithms were evaluated within a gender adaptation framework, using the TIMIT dataset. The TIMIT dataset has an imbalance in the number of male and female speakers, and the male training set was therefore used as the prior dataset and the female dataset as the adaptation set. A smaller female dataset was also created, so as to determine the effect of the algorithms when an even larger imbalance exists. Two criteria were used to evaluate the algorithms, namely adaptation performance and training performance. The female testing set accuracy and the minimum of the female and male testing set accuracies were used to determine the performance for these two criteria.

Here, the MAPMCE algorithm performed best for both datasets and both testing criteria, resulting in marked improvements in accuracy. The MAP and GMAP algorithms resulted in little or no improvement in accuracy for the female testing set (adaptation). Reasonable improvements in the minimum testing set accuracy were attained by these two algorithms when the small female training set is used, with the GMAP algorithm performing slightly better than the standard MAP algorithm.

In Section 4.6.3 the algorithms were tested within a gender adaptation framework for a connected digit task. The TIDIGIT database was used for this purpose. Here, once again the MAPMCE algorithm worked best, followed by the GMAP algorithm and the MAP algorithm performing worst. However, when duration modeling is used, the GMAP algorithm results in the best performance, with relative improvements in digit accuracy of up to 54.9%.

When comparing MCE or MAPMCE to non-discriminative algorithms such as MAP and GMAP, one must be careful as they are often not directly comparable. MCE as mentioned, due to its discriminative nature, automatically tends to incorporate duration and language modeling information into the HMMs. This means that when duration modeling or language modeling is used, the effect thereof will be smaller when using MCE. McDermott [75] noted this, when he used a bigram language model with MCE and compared it to using a bigram model with ML trained HMMs.

Table 4.23 gives the execution times for the three algorithms (MAP, GMAP and MAPMCE) on a Pentium III 600MHz computer. The times given are for one iteration only; total execution time is equal to execution time given in Table 4.23 times the number of iterations used. Given that the GMAP and MAPMCE algorithms required more iterations (around 30) than that required by MAP (10 iterations), it is evident that the GMAP and MAPMCE algorithms are considerably more computationally expensive than the MAP algorithm. The improved performance, however, warrants the additional computational expense.

Table 4.23: Execution times (in minutes) for one iteration (update) of the three algorithms (MAP, GMAP and MAPMCE) on a Pentium III 600MHz computer. The prior data set (square brackets) and adaptation set (round brackets) are included in the dataset description.

Dataset	MAP	GMAP	MAPMCE
SUNSpeech (A) $[E]$	1.3	13.4	37.7
TIMIT (T_F) $[T_M]$	4.2	16.9	54.1
TIDIGITS (T_W) $[T_M]$	0.3	4.8	18.0

4.7 Summary

This chapter introduced Bayesian adaption and its usage within a continuous speech recognition framework. The MAP algorithm of Gauvain and Lee [45] was described. A gradient-based MAP algorithm which makes no assumption about the form of the prior was proposed and the implementation thereof was discussed. A Bayesian inspired MCE-based adaptation algorithm (MAPMCE) was also proposed. The MAPMCE algorithm is an extension of the MCE algorithm and the implementation thereof is relatively simple.

The three algorithms were experimentally evaluated in Section 4.6, using the SUN-Speech database for language adaptation, and the TIMIT and TIDIGIT databases for gender adaptation experiments. On the whole, the MAPMCE algorithm proved to work best, with the GMAP algorithm performing reasonably well. The MAP algorithm, in general, resulted in considerably worse performance than either the GMAP or MAPMCE algorithms.

Chapter 5

Bayesian learning

This chapter develops a Bayesian approach to learning for HMMs in speech recognition. Markov chain Monte Carlo methods which can be used to numerically integrate the posterior distribution as required by the Bayesian learning approach form part of this discussion. The implementation of Bayesian learning for HMMs in speech recognition is presented, including the requirement of maintaining the original HMM constraints, choice of prior and utterance recognition. This work shows that the Bayesian learning approach can be successfully applied to complex models when the amount of training data is small. This is contrary to the notion that one must limit the complexity of the model when training data is limited, as was discussed in Section 2.2.1.

This work was inspired by the work of Neal [82], who proposed a Markov chain Monte Carlo based Bayesian learning procedure for neural networks. In this chapter, the Bayesian learning procedure used by Neal will be implemented and adapted for usage with hidden Markov models and speech recognition. Previous applications of Bayesian learning in speech recognition have concentrated on using approximations. One such approximation is that of Huo *et al.* [54, 58] who used a Gaussian distribution to approximate the posterior distribution. The MAP estimation algorithm (Chapter 4) can also be regarded as an approximate Bayesian approach. However, my formulation

is the first Markov chain Monte Carlo based Bayesian learning approach used for hidden Markov model speech recognition systems.

The Bayesian learning framework is introduced here and past work both within the field of speech recognition and in the more general field of neural networks is discussed.

5.1 Introduction

Bayesian methods can be used for the inference of parameter values in a model given the data. Bayesian methods have also been used for the purpose of model comparison. David Mackay [72] focused primarily on the usage of Bayesian methods for the comparison and training of neural networks. Most people would include the above two uses of Bayesian methods in the data modeling process.

The remainder of this section will summarize the relevant Bayesian theory used in this chapter. Certain sections from Chapter 4 have been reproduced for readability. For a more complete introductory text on Bayesian statistics, the reader is referred to Box and Tiao [13], DeGroot [27] and Bishop [12]. The theory and discussions in this section will be biased towards speech recognition applications of Bayesian learning.

5.1.1 Bayes' theorem

The fundamental concept of Bayesian analysis is that the plausibilities of alternative hypotheses are represented by probabilities, with inference being performed by evaluating these probabilities.

Given a vector $\mathbf{y} = (y_1, \ldots, y_n)$ of n observations, with probability distribution $P(\mathbf{y}|\theta)$, which depends on the k parameters $\theta^T = (\theta_1, \ldots, \theta_k)$. The parameter vector θ has the probability distribution $P(\theta)$. Given the observed data, the conditional distribution of

θ is

$$P(\theta|\mathbf{y}) = \frac{P(\mathbf{y}|\theta)P(\theta)}{P(\mathbf{y})}. \tag{5.1}$$

The denominator in Eq. (5.1), $P(\mathbf{y})$, is a normalizing factor, which ensures that the integral of $P(\theta|\mathbf{y})$ is equal to one. It can be written as follows:

$$P(\mathbf{y}) = \int P(\mathbf{y}|\theta)P(\theta)d\theta. \tag{5.2}$$

Equation (5.1) is referred to as Bayes' theorem. The distribution $P(\theta)$, is called the *prior* distribution and expresses what is known about the model parameters before any data is observed. The *posterior* distribution $P(\theta|\mathbf{y})$, tells us what is known about the model parameters given that data has been observed. In what follows, the prior distribution and posterior distribution will again sometimes simply be referred to as the "prior" and "posterior" respectively.

The distribution $P(\mathbf{y}|\theta)$ is often referred to as the data *likelihood* and can be written $L(\theta|\mathbf{y})$. This is valid when $P(\mathbf{y}|\theta)$ is regarded as a function of $\mathbf{y}$ and not of θ.

In many Bayesian methods, the normalizing constant is not necessary and Eq. (5.1) is written as

$$P(\theta|\mathbf{y}) \propto L(\theta|\mathbf{y})P(\theta). \tag{5.3}$$

5.1.2 Bayesian learning and prediction

The result of Bayesian learning is a probability distribution (posterior) which expresses our beliefs of how likely individual parameters values are. This is the basis for Bayesian

learning, as it allows learning to be performed using probability theory.

In a Bayesian approach to HMM parameter estimation and recognition, the objective is to find a predictive distribution for an unknown utterance, given the observations of the utterance, as well as the training observations. Let the observations for the ith utterance be written as $\mathbf{O}_i$. For n training utterance examples $\mathbf{O} = (\mathbf{O}_1, \ldots, \mathbf{O}_n)$, Bayes' theorem (Eq. (4.1)) can be written as

$$\begin{aligned} P(\theta|\mathbf{O}) &= \frac{P(\mathbf{O}|\theta)P(\theta)}{P(\mathbf{O})} \\ &\propto P(\mathbf{O}|\theta)P(\theta). \end{aligned} \tag{5.4}$$

Assuming independence of the observations we can write the likelihood as follows:

$$P(\mathbf{O}|\theta) = \prod_{i=1}^{n} P(\mathbf{O}_i|\theta). \tag{5.5}$$

In a Bayesian framework, when we wish to classify an unknown input, we need to calculate the following probability,

$$P(\mathbf{O}_{unknown}|\mathbf{O}_1^{(i)}, \ldots, \mathbf{O}_n^{(i)}) = \int P(\mathbf{O}_{unknown}|\theta)P(\theta|\mathbf{O}_1^{(i)}, \ldots, \mathbf{O}_n^{(i)})d\theta, \tag{5.6}$$

where i is the class and $\mathbf{O}_{unknown}$ is the unknown observation. The classifier decision C is the class resulting in the highest value of Eq. (5.6), i.e.

$$C(\mathbf{O}_{unknown}) = i \quad \text{where} \quad i = \underset{j}{argmax} P(\mathbf{O}_{unknown}|\mathbf{O}_1^{(j)}, \ldots, \mathbf{O}_n^{(j)}), \tag{5.7}$$

where $C(\mathbf{O}_{unknown})$ is the classifier's decision for the unknown observation.

Unfortunately, due to the nature of the incomplete data problem caused by the underlying hidden processes of an HMM, the evaluation of Eq. (5.6) is non-trivial. If, however, the posterior ($P(\theta|\mathbf{O})$) is well approximated by a Gaussian, then Eq. (5.6) can be approximated as follows [72]:

$$P(\mathbf{O}_{unknown}|\mathbf{O}_1^{(i)},\ldots,\mathbf{O}_n^{(i)}) \approx P(\mathbf{O}_{unknown}|\theta_{MAP})P(\theta_{MAP})(2\pi)^{D/2}|A|^{1/2} \tag{5.8}$$

where θ is D-dimensional, θ_{MAP} is the maximum *a-posteriori* point (mode of the posterior $P(\theta|\mathbf{O})$) and A is the modal dispersion matrix, i.e., $A = -V^{-1}$, where V is the Hessian matrix of second derivatives of the log of the posterior evaluated at the mode of the posterior.

This approximation has been used extensively in Bayesian approaches. MacKay [72] used this approximation in his work on model selection for neural networks. Huo *et al.* [54, 58] proposed a *quasi-Bayesian predictive classification* approach for continuous density HMMs, in which they used this approximation. Another approximation was also proposed by Huo *et al.* [57], which used the following Viterbi-based approximation,

$$P(\mathbf{O}_{unknown}|\mathbf{O}_1,\ldots,\mathbf{O}_n) \approx \max_{\mathbf{q},\mathbf{l}} \int P(\mathbf{O}_{unknown}|\theta)P(\theta|\mathbf{O}_1,\ldots,\mathbf{O}_n)d\theta, \tag{5.9}$$

where $\mathbf{q}$ is a state sequence and $\mathbf{l}$ is the sequence of associated mixture components. A modified Viterbi algorithm was used to compute the above approximation.

We can, however, use Monte Carlo (MC) methods to obtain a better approximation of Eq. (5.6) than Eq. (5.8) or Eq. (5.9). MC methods make no assumption concerning the form of the distribution, as done in the above approximations. In theory, MC methods can approximate Eq. (5.6) for complex distributions with multiple modes, as well as distributions for which the dominant contribution of the integral results from areas in parameter space which are not near a mode. Markov chain Monte Carlo methods [12, 82] will therefore be used in this implementation and will be described in Section

5.2.

In the field of neural networks, Neal [82] used the "Hybrid Monte Carlo" method (described later in Section 5.2.3) for Bayesian learning. The following are some of the applications using Monte Carlo algorithms that have been reported for speech recognition or speech processing. Vermaak and Niranjan [114] used a Markov chain Monte Carlo algorithm for speech enhancement. Robert *et al.* [98] presented a Markov chain Monte Carlo strategy to obtain a marginal MAP estimate. Godsill and Andrieu [48] used Markov chain Monte Carlo methods for the separation and recovery of convolutively mixed autoregressive processes. We will, however, use Markov chain MC methods to implement Bayesian learning for HMM speech recognizers.

5.1.3 Bias/variance problem

Let us once again look at the bias/variance problem discussed in Section 2.2.1 as it is central to the Bayesian learning approach. Integrating over the posterior, as in Eq. (5.6), results in the variance term of Eq. (2.25) being greatly reduced. Adjusting the complexity of models based on the amount of training data in a Bayesian framework therefore makes little sense as the variance term effectively disappears (except for extreme sparse training data). More complex models, which perform worse when using a single point estimate (ML), will therefore perform better than less complex models in a Bayesian implementation. We will, however, prefer simpler models due to other reasons, such as computational complexity.

Numerical integration with respect to the posterior $P(\theta|\mathbf{O})$ using a fixed number of samples N will, however, increase the variance of the solution. The effect of the number of samples used will be investigated in the experimental section later in this chapter.

5.1.4 Hierarchical models

Hidden Markov models are relatively complex and have many parameters to estimate. It is, therefore useful to specify the joint distribution of some of these parameters in terms of a common hyperparameter γ which has a prior distribution of its own. This is known as a *hierarchical model.*

The prior distribution $P(\theta)$ can then be written in terms of the hyperparameters as follows (assuming independence),

$$P(\theta) = \int P(\gamma) \prod_{i=1}^{D} P(\theta_i|\gamma) d\gamma \tag{5.10}$$

where $P(\gamma)$ is the prior distribution of the hyperparameter γ, $P(\theta_i|\gamma)$ is the prior distribution of the parameter θ_i given the hyperparameter.

A hierarchical model, if well formulated, can be considerably more intelligible than using a direct prior distribution. We can also in this way, incorporate vague heuristic information into the prior, as will be done in Section 5.3.

5.2 Monte Carlo methods

As mentioned in Section 5.1.2, we want to evaluate Eq. (5.6), which is the expectation of the function $P(\mathbf{O}_{unknown}|\theta)$ with respect to the posterior distribution $P(\theta|\mathbf{O}_1, \ldots, \mathbf{O}_n)$. Such expectations can be estimated using Monte Carlo methods, by summing $P(\mathbf{O}_{unknown}|\theta^{(j)})$ using N samples of θ (the ith sample denoted by $\theta^{(i)}$) generated from the posterior distribution $P(\theta|\mathbf{O})$ for $j = \{1, \ldots, N\}$, i.e.

$$P(\mathbf{O}_{unknown}|\mathbf{O}_1, \ldots, \mathbf{O}_n) \approx \sum_{j=1}^{N} P(\mathbf{O}_{unknown}|\theta^{(j)}), \tag{5.11}$$

where the samples $\theta_1, \ldots, \theta_N$ are generated by a process such that the distribution thereof is that of the posterior $P(\theta|\mathbf{O})$.

The sampling methods described in this section were developed for situations where probability distribution cannot be directly sampled. Sampling from a one-dimensional Gaussian distribution can, for example, be done directly.

Suppose we wish to generate a sample from a distribution $P(\theta)$ for $\theta \in \Theta$, but cannot do so directly. This can be done by constructing a Markov chain with state space Θ, which is easy to simulate, and whose equilibrium distribution is $P(\theta)$.

The following are sufficient conditions for such an algorithm to approach the desired distribution [111]:

- Invariance with respect to the distribution P. If for all pairs of configurations θ and θ',

$$\frac{P(\theta \to \theta')}{P(\theta' \to \theta)} = \frac{P_\infty(\theta')}{P_\infty(\theta)} \tag{5.12}$$

  and at step n we have $P_n(\theta) = P_\infty(\theta)$, then at step $n+1$ we will have $P_{n+1}(\theta) = P_\infty(\theta)$. The desired distribution is therefore an equilibrium distribution. This condition is called the *detailed balance* condition, and any chain which satisfies it is said to be reversible. The resulting distribution $P(\theta)$ persists once established and is therefore invariant (or stationary).

- Ergodicity. This condition specifies that the probability distribution at step $n+1$ should be closer to $P_\infty(\theta)$ than at step n. An algorithm which complies with condition will converge to its equilibrium condition from any initial configuration.

In the following sections, three Markov chain Monte Carlo methods will be described. The above conditions (detailed balance and ergodicity) will be used to determine the suitability of each method for the implementation of Bayesian learning. We will in

particular find that *Gibbs sampling* (Section 5.2.1) is not ergodic and will therefore not be used (directly) to sample the posterior distribution $P(\theta|\mathbf{O})$ of an HMM. The *stochastic dynamics* (Section 5.2.2) and *hybrid Monte Carlo* (Section 5.2.3) methods meet both of the above conditions and we will therefore be able to use either of these procedures.

5.2.1 Gibbs sampling

Gibbs sampling [82, 7] can be used to sample the distribution of a multi-dimensional parameter. Gibbs sampling is also known as the *heatbath* method in the physics literature.

In Gibbs sampling a Markov chain is simulated, in which the new n-dimensional sample $\theta^{(t+1)}$ is generated from $\theta^{(t)}$ using the following iterative procedure:

Generate $\theta_1^{(t+1)}$ from the conditional distribution of θ_1 given $\theta_2^{(t)}, \theta_3^{(t)} \ldots, \theta_n^{(t)}$.
Generate $\theta_2^{(t+1)}$ from the conditional distribution of θ_2 given $\theta_1^{(t+1)}, \theta_3^{(t)} \ldots, \theta_n^{(t)}$.
$\vdots$
Generate $\theta_i^{(t+1)}$ from the conditional distribution of θ_i given $\theta_1^{(t+1)}, \ldots \theta_{i-1}^{(t+1)}, \theta_{i+1}^{(t)} \ldots, \theta_n^{(t)}$.
$\vdots$
Generate $\theta_n^{(t+1)}$ from the conditional distribution of θ_n given $\theta_1^{(t+1)}, \ldots\ldots, \theta_{n-1}^{(t+1)}$.
Generate $\theta_1^{(t+2)}$ from the conditional distribution of θ_1 given $\theta_2^{(t+1)}, \theta_3^{(t+1)} \ldots, \theta_n^{(t+1)}$.
$\vdots$
and so on.

The transition resulting from the above steps being executed leaves the desired distribution Q invariant and is reversible. Gibbs sampling is, however, not necessarily an ergodic Markov chain and depending on the situation will not converge to its equilibrium distribution if we do not start from the desired distribution.

Djurić and Chun [34] proposed a method of estimating non-stationary (duration modeling) discrete hidden Markov models. The posterior of the model parameters was sampled using a Gibbs sampler. Their implementation was tested on an extremely simple HMM consisting of 3 states and with 5 possible emission variables. Histograms of the posterior samples for certain parameters were presented, showing that the mode of the posterior was reasonably accurate.

Gibbs sampling is dependent on being able to sample the distribution of one parameter conditional on the other parameters. For continuous density HMMs, the conditional distribution of one parameter given the values of the other parameters is non-trivial, where any parameter of a given state is dependent on all the other parameters of all the states. Gibbs sampling is, therefore, not a suitable sampling method that can be used to sample the posterior of an HMM. The posterior of HMMs will be multi-modal and due to the fact that Gibbs sampling is not ergodic, it will also not necessarily converge to the correct distribution under these conditions.

We will not use Gibbs sampling in itself, but rather as part of the stochastic dynamics and hybrid Monte Carlo algorithms described later in this section.

5.2.2 The stochastic dynamics method

The *stochastic dynamics method* [82] for sampling of distributions, otherwise known as the *refreshed molecular dynamics method*, was introduced by Anderson [3] and applied to the field of quantum chromodynamics by Duane and Kogut [36, 38]. It is also sometimes known in the literature as the *hybrid* method, because it contains two standard update steps – uniformly sampling values of variables q and p which have a fixed total energy $H(q,p)$, and sampling states with different values of H. Here, the stochastic dynamics method is used to generate the samples of the posterior distribution $P(\theta|\mathbf{O})$, which will be used to numerically integrate Eq. (5.6).

Let us assume that we wish to sample from a distribution for a variable q, which has n dimensions. In the systems for which the techniques described in this section were developed, q is typically the coordinates of the particles in a physical system. In our work, q will be the HMM parameters, i.e. $q = \theta$. We therefore have a system with continuously valued coordinates q_i, with the probability of the variable q defined as

$$P(q) \propto e^{(-E(q))}, \tag{5.13}$$

where $E(q)$ is the potential energy function. Any non-zero probability function, can be written in this form by defining $E(q) = -ln[P(q)] - ln(Z)$ for $Z > 0$.

A *momentum* variable p is introduced which has n components, one for each of the components of q. Here, the kinetic energy is used which is

$$K(p) = \sum_{i=1}^{n} \frac{p_i^2}{2m_i}, \tag{5.14}$$

with m_i being the "mass" associated with each component. It will be assumed, for the rest of this discussion, that $m_i = 1$ for all parameters. The terms *state* and *configuration* will be used to indicate the combination of the system coordinates ($q = \theta$) and momenta (p), i.e. $state = configuration = (p, q)$. The total energy is $H(p, q) = E(q) + K(p)$ and the probability for q and p is therefore

$$P(p, q) \propto e^{-H(p,q)}. \tag{5.15}$$

Consider Hamilton's equations of motion for this system,

$$\frac{\partial q_i}{\partial t} = \frac{\partial H}{\partial p_i} = p_i \tag{5.16}$$

$$\frac{\partial p_i}{\partial t} = -\frac{\partial H}{\partial q_i} = -\frac{\partial E}{\partial q_i}. \tag{5.17}$$

Three important properties of the Hamiltonian dynamics necessary for sampling are:

- The motion defined by the above equations leave the total energy H constant.
- The volume of regions of phase space is conserved, i.e. if we follow points in some region of volume V, we find that the region where these points end up after a given time T also has volume V. This is important as the probability of a configuration is really the probability density times the volume element in phase space.
- The motion is reversible. After having simulated Hamilton's equations for a time T, we can change the sign of the momenta, apply Hamilton's equations for the same period of time and end at the original starting point.

An update which consists of generating a random number R, multiplying all the momenta by -1 if $R < \frac{1}{2}$ and then integrating Hamilton's equations for a given time interval satisfies the conditions required by the fundamental theorem [111].

Since each p_i is independent of the q_i and the other p_i, the probability distribution of the momenta can be sampled by using Gibbs sampling and assuming each to be a Gaussian distributed random number. Note that, it is not necessary to randomly reverse the momenta as described above, as the Gibbs sampling is just as likely to generate p_i as $-p_i$. This is known as "refreshing" the momentum variable p.

The length of time T over which Hamilton's equations are integrated is a critical parameter which must be found. Figure 5.1 illustrates the path of the stochastic dynamic method for different integration times. If the time T is small, the coordinates

q will not change much and the result is effectively a random walk of parameter space (Figure 5.1 (a)). Alternately, a large integration time T will result in a path which is periodic in nature and we will waste our time in generating such a long path which could easily end up close to where we started (Figure 5.1 (b)). There is therefore an optimal value of T which lies somewhere between the two extremes (Figure 5.1 (c)). In certain situations this can be done analytically [39], but in realistic systems the simulation time T must be empirically determined. Fortunately, however, there is often a relatively large range of T which give good results. Ultimately, we wish to ensure that the correlation between updates is a minimum.

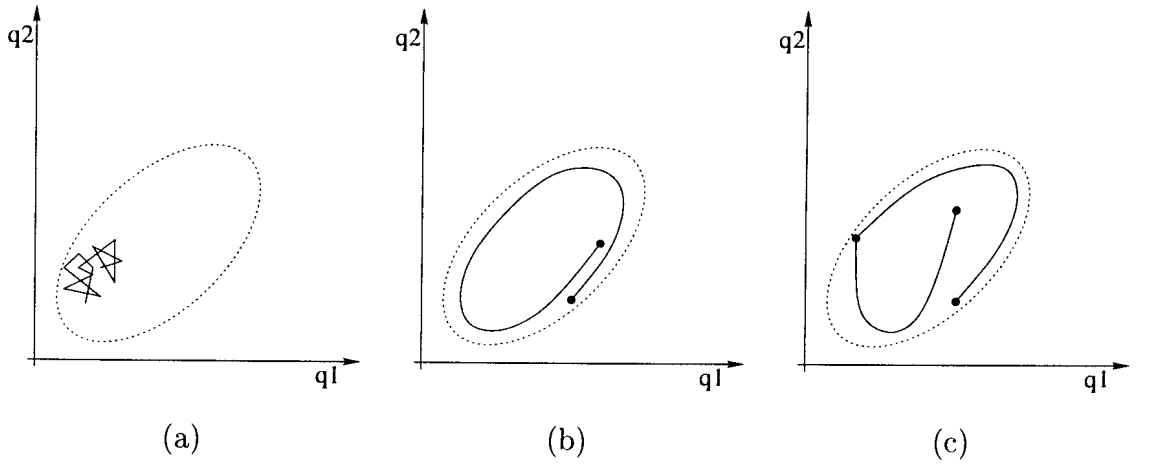


Figure 5.1: Movement through parameter space using the stochastic dynamics sampling method. The underlying distribution being sampled is indicated using a dotted line. (a) Small time period T; a random walk of parameter space. (b) Large time period T; a waste of time in "periodic" movement. (c) More optimal simulation time T.

In practice, we cannot integrate the Hamiltonian dynamics exactly. The leapfrog integration scheme described next can, however, be used to approximate the Hamiltonian dynamics.

Leapfrog integration In the leapfrog integration scheme [111] approximations of the position and momentum, $q_i(t+\epsilon)$ and $p_i(t+\epsilon)$ from $q_i(t)$ and $p_i(t)$ are obtained as follows:

$$p_i(t+\frac{\epsilon}{2}) = p_i(t) - \frac{\epsilon}{2}\frac{\partial E(t)}{\partial q_i} \tag{5.18}$$

$$q_i(t+\epsilon) = q_i(t) + \epsilon\frac{p_i(t+\frac{\epsilon}{2})}{m_i} \tag{5.19}$$

$$p_i(t+\epsilon) = p_i(t+\frac{\epsilon}{2}) - \frac{\epsilon}{2}\frac{\partial E(t+\epsilon)}{\partial q_i}, \tag{5.20}$$

where the step size ϵ is a small, finite, positive constant.

In order to follow the Hamiltonian dynamics for a given time T, Eqs. (5.18) to (5.20) are applied in order for $L = \frac{T}{\epsilon}$ steps. When applying the leapfrog step more than once, the last momentum update (Eq. (5.20)) and the first (Eq. (5.18)) of the next step can be combined. All but the very first and very last momentum half-steps can be merged. Figure 5.2 illustrates the leapfrog integration scheme of q and p over a time interval $[0, T]$ using the energy and momentum steps defined in Eqs. (5.18) to (5.20). If preferred, Eqs. (5.18) to (5.20) can be rewritten such that we start with an energy half-step followed by a full momentum step.

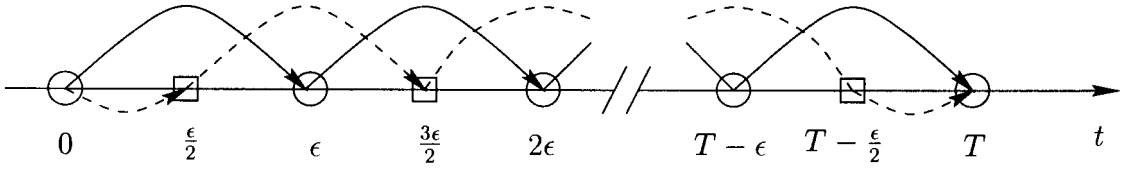


Figure 5.2: The leap frog integration scheme of q and p over a time interval $[0, T]$. The position q is evaluated at the points marked with circles and the momentum p at the points marked with squares. The dashed line indicates the momentum update and the solid line indicates the energy update.

Each of the energy and momentum steps has an error of order ϵ^3 [111]. Integrating for L leapfrog steps will therefore result in an error of order $L\epsilon^3$. A step size ϵ is chosen which is small enough to give an acceptable error. If we choose $L \approx \frac{1}{\epsilon}$, which is often done, then the error is of order ϵ^2. The hybrid Monte Carlo algorithm discussed next, is based on the stochastic dynamics method and was developed to eliminate these systematic errors introduced by leapfrog integration.

5.2.3 The hybrid Monte Carlo algorithm

The hybrid Monte Carlo algorithm of Duane *et al.* [37] for sampling of distributions eliminates the systematic errors of the stochastic dynamics method resulting from the finite integration step size, where the total energy is not conserved during leapfrog integration. The Hybrid Monte Carlo algorithm is an extension of the stochastic dynamics method and uses the Metropolis algorithm [79] to eliminate the bias introduced by the errors resulting from the leapfrog integration of the Hamiltonian dynamics.

The Metropolis algorithm The Metropolis algorithm was introduced in 1953 by Metropolis *et al.* [79]. It has since been extensively used, and is the basis for the simulated annealing optimization method: Letting A be the current configuration and B a potential configuration (q, p) generated such that $P(A \rightarrow B) = P(B \rightarrow A)$, where $P(A \rightarrow B)$ is the probability of generating the trial configuration B given the current configuration A. If $P(B) > P(A)$ then configuration B is accepted. If $P(B) \leq P(A)$, then the configuration B is accepted with probability $\frac{P(B)}{P(A)}$. If B is rejected, the next configuration is A.

The hybrid Monte Carlo algorithm uses the Metropolis algorithm to determine whether or not to accept a new configuration generated using the refresh and integration step of the stochastic dynamics method. A hybrid Monte Carlo algorithm step can therefore be described as follows:

1. Refresh the momenta using Gibbs sampling.
2. Starting with the current state (p, q), perform L leapfrog steps to generate a trial next configuration (p', q').
3. Accept the trial next configuration with probability $min(1, exp(H(p, q) - H(p', q'))$, otherwise choose the new state to be the same as the old.

If Hamilton's equations were simulated exactly, the change in H would be zero and the trial configuration would always be accepted. When an approximation is used (leapfrog), H will change and a trial configuration will sometimes be rejected. This exactly eliminates the bias introduced by leapfrog. Note that it is important to maintain a relatively high acceptance rate, so as to minimize correlation between consecutive states.

It is expected that the work required in simulating Hamilton's equations for a fixed time will grow with the volume (V) of the system as $T \propto V^{\frac{5}{4}}$ [26, 50], as compared to growth proportional to V for the stochastic dynamics method. Although this is a relatively slow growth, as pointed out by Toussaint [111], the stochastic dynamics method will be eventually be preferred for systems which are very large. Due to current computational resources, and the large size of HMM systems, the hybrid Monte Carlo algorithm turned out to be infeasible for our work. This will, however, undoubtedly change as computers become faster with time.

We have, up until now, only investigated the theoretical aspects of implementing Bayesian learning. The next section will discuss implementation issues for a Bayesian learning approach for HMM speech recognizers.

5.3 Implementation of Bayesian HMM learning

The training process for the implementation of Bayesian learning described in this chapter generates the N_m samples of the posterior $P(\theta|\mathbf{O})$ required to numerically integrate Eq. (5.6) during recognition of an unknown utterance. This section will discuss several aspects of the implementation for the training and recognition processes.

The prior used here is the same as in the gradient-based MAP algorithm described in Section 4.4. The gradient of the energy function $E(\theta)$ is required here for the leapfrog integration. This gradient (derivative) of $E(\theta)$ with respect to the individual

parameters of both HMM and prior are therefore exactly the same as in Chapter 4 (Eq. (4.47) to Eq. (4.64)) and are as a result, not reproduced here.

5.3.1 HMM constraints

In Chapter 4 we used transformations to ensure that the original HMM constraints were maintained. Unfortunately, we cannot use transformations when we wish to sample a distribution. These are second order sampling techniques and using transforms will deform the resultant distribution being sampled. To understand this, let us look at a simple example.

Let us assume that we wish to sample the distribution of the standard deviation σ of a Gaussian model, and that the gradient of the energy function with respect to σ is a constant (k) for a given region in parameter space (i.e. $\frac{\partial E}{\partial \sigma} = k$). Here, the coordinate variable q consists of only the variable σ, i.e. $q = \{\sigma\}$. Given the momentum at time t, $p(t)$, a single leap frog momentum step (Eqs. (5.18) to (5.20)) will be

$$\begin{aligned} p(t + \frac{\epsilon}{2}) &= p(t - \frac{\epsilon}{2}) - \epsilon k, \\ q(t + \epsilon) &= q(t) + \epsilon p(t + \frac{\epsilon}{2}). \end{aligned} \tag{5.21}$$

We would therefore expect the coordinate variable q to accelerate at a constant rate ($\propto -\epsilon k$) in this region of parameter space. If, however, we use the following transform $\tilde{\sigma} = \ln \sigma$, then the derivative of E with respect to $\tilde{\sigma}$ becomes $k\sigma$ and the leapfrog step becomes

$$\begin{aligned} \tilde{p}(t + \frac{\epsilon}{2}) &= \tilde{p}(t - \frac{\epsilon}{2}) - \epsilon k\sigma \\ \tilde{q}(t + \epsilon) &= \tilde{q}(t) + \epsilon \tilde{p}(t + \frac{\epsilon}{2}). \end{aligned} \tag{5.22}$$

Note that $\tilde{p}(t) = p(t)\frac{\partial\tilde{\sigma}}{\partial\sigma}$. Transforming these update equations back from the transformed domain results in the following update

$$\begin{aligned} p(t+\frac{\epsilon}{2}) &= p(t-\frac{\epsilon}{2}) - \epsilon k\sigma^2, \\ q(t+\epsilon) &= q(t)e^{\epsilon p(t+\frac{\epsilon}{2})/\sigma}. \end{aligned} \tag{5.23}$$

Note that neither the coordinate update $q(t+\epsilon)$ or momentum update $p(t+\frac{\epsilon}{2})$ are as in Eq. (5.21), as they should be. The distribution sampled will therefore be a distorted version of the true distribution.

Transformations were used in the gradient based MAP algorithm (GMAP) as there we were only searching for the mode of the posterior and the position thereof will not change when using transformations. But now it is not possible to use transformations with the Markov chain Monte Carlo methods described. A constrained version of the sampling algorithms is therefore implemented, in which the HMM constraints are applied. In particular, the variances are not allowed to be below a predefined value which is greater than zero (0.0001 has been used throughout).

It is non-trivial (if not impossible) to implement a constrained algorithm which maintains the constraints for the transition probabilities and mixture weights (i.e. $\sum_j a_{ij} = 1$ and $\sum_i c_i = 1$), without using transformations. These parameters are therefore treated as being fixed (at some estimate such as ML) and only the posterior distribution of the means and variances of the HMM Gaussian mixtures are therefore sampled.

5.3.2 HMM prior and hyperparameters

The prior distribution for the HMM Gaussian mixture mean and variances is again chosen to be a normal-Wishart distribution. Prior distributions are not required for the mixture weights or transition probabilities, as they are assumed to be fixed (as

motivated above) and their posterior is not sampled. The normal-Wishart prior was introduced in Section 4.2 and is reproduced here for convenience,

$$g_{gaussian}(r_{jk}, \mu_{jk}|n_{jk}, \nu_{jk}, m_{jk}, \tau_{jk}) = (2\pi)^{-\frac{D}{2}}|\nu_{jk}r_{jk}|^{\frac{1}{2}}e^{-\frac{1}{2}\nu_{jk}(\mu_{jk}-m_{jk})^T r_{jk}(\mu_{jk}-m_{jk})} c|\tau_{jk}|^{-\frac{n_{jk}}{2}}|r_{jk}|^{\frac{(n-D-1)}{2}}e^{-\frac{1}{2}tr(r_{jk}\tau_{jk}^{-1})}, \quad (5.24)$$

where $(n_{jk}, \nu_{jk}, m_{jk}, \tau_{jk})$ are the prior distribution parameters associated with mixture component k of state j. See Table 4.1 for a summary of the HMM parameters and their associated priors and prior parameters. The value c is a normalizing constant which ensures that the integral of the prior is equal to one.

It can be easily shown that choosing the parameters m_{jk} and τ_{jk} in Eq. (5.24) to be

$$m_{jk} = \mu' \quad (5.25)$$

$$\tau_{jk} = (n_{jk} - D)\Sigma', \quad (5.26)$$

results in the mode of the prior being at the point $\mu_{jk} = \mu'$ and $\Sigma_{jk} = \Sigma'$. The parameters n_{jk} and ν_{jk} determine the degree to which the prior is peaked about its mode.

Although not necessary, it makes sense (simplifies calculations) to reduce the number of variables in the prior by expressing the parameters n_{jk} and ν_{jk} in terms of a common parameter C_{jk},

$$n_{jk} = C_{jk} + D \quad (5.27)$$

$$\nu_{jk} = C_{jk} + 1, \quad (5.28)$$

with $C_{jk} > 0$.

The parameters C_{jk}, m_{jk} and τ_{jk} are now given their own distributions. These distributions and their parameters must be chosen in a meaningful way, such that the hyperparameters contain *a-priori* knowledge (albeit vague). The choice of the distributions and parameters are discussed next.

We would expect the parameters for the mixtures of a given state to be highly correlated. This is often not the case with HMMs, where a single state can represent multiple acoustic units, which can result in a higher confusability.

Let us, for example, look at a three state HMM, with each state having several Gaussian mixtures, used to represent the word "four". If, assuming, certain of the training utterances are pronounced such that the "r" is not heard, then the last state will probably represent both the consonant *r* and the vowel *ou* which is the end of some of the training utterances. The result is a model which is easier to confuse with other utterances (or part thereof). For example, this HMM will give a high output for the phone sequence *f-ou-r-ou-r*.

Illina and Gong [55] investigated this phenomenon, which they called trajectory folding. One solution to this problem is to change the topology of the model. It was found that using a trajectory mixture HMM (TMHMM) [107] and segment-based mixture stochastic trajectory HMMs (MSTM) [49] reduced the effect of this problem.

Trajectory folding, is not only caused by differences in pronunciation, but can also be caused by the training procedure. This is especially the case when the model is complex and little training data is available. From the above discussion, it is evident that states representing multiple acoustic units are not desirable. In an attempt to avoid trajectory folding, the prior parameters of a given state have therefore been given common distributions and hyperparameters. Note that the prior parameter distributions and the associated hyperparameters which have been chosen are not necessarily the only (or best) possible choice. The manner in which the parameters are grouped is a modeling choice, which is made on the basis of prior knowledge. Many possibilities

remain, and it is more than likely that the performances reported in this chapter can be improved upon by making other choices.

Our choice of the priors are as follows:

The prior mean m_{jk} of state j is given a normal distribution with mean ω_j and standard deviation ς_j, i.e.

$$P(m_{jkl}) = \mathcal{N}(\omega_{jl}, \varsigma_{jl}), \quad k = 1, \ldots, M, \tag{5.29}$$

where M is the number of mixture components, and ω_{jl} and ς_{jl} are the hyperparameters for the prior mean m_{jk}.

Given that we wish to keep the hyperparameter distributions as intuitive as possible, the distribution of the prior mode Σ' in Eq. (5.26) has been chosen to be a gamma distribution (Section A.4) with parameters ϕ_j and ψ_j. The distribution of the prior parameter τ_{jk} (precision of μ_{jk}) of state j can, using Eq. (5.26), therefore be written as

$$P(\tau_{jkl}) = (n_{jk} - D)\mathcal{G}(\phi_{jl}, \psi_{jl}), \quad k = 1, \ldots, M, \tag{5.30}$$

where ϕ_{jl} and ψ_{jl} are the hyperparameters of τ_{jkl}. Note that from Eq. (5.25), the prior mean m_{jk} is also the mode of the prior with respect to the mean parameters. The distribution for the prior parameter C_{jk} is also chosen to be a gamma distribution with parameters υ and ϱ, i.e.

$$P(C_{jk}^{(i)}) = \mathcal{G}(C_m, C_v), \quad k = 1, \ldots, M, \tag{5.31}$$

for every state $j = 1, \ldots, N$ in every HMM $i = 1, \ldots, N_h$. A summary of the HMM

parameters, their prior distributions, the prior parameters and their distributions can be found in Table 5.1.

Table 5.1: Summary of the HMM parameters, prior distributions, hyperparameters and their distributions. A Wishart distribution is represented with $\mathcal{W}$ and a Gamma with $\mathcal{G}$.

Parameter	Prior distribution	Prior parameter	Distribution
$\mu_{jk}\|R_{jk} = r_{jk}$	$\mathcal{N}(m_{jk}, (C_{jk}+1)r_{jk})^{-1/2})$	m_{jk}	$\mathcal{N}(\omega_{jl}, \varsigma_{jl})$
R_{jk} (Σ_{jk}^{-1})	$\mathcal{W}((C_{jk}+D), \tau_{jk})$	C_{jk}	$\mathcal{G}(\upsilon, \varrho)$
		τ_{jk}	$(n_{jk}-D)\mathcal{G}(\phi_{jl}, \psi_{jl})$

Determining hyperparameters The ML estimate, which is used as the starting point for the stochastic dynamics sampling, is used to estimate the hyperparameters ω_j, ς_j (prior mean m_{jk} in Eq. (5.29)) and ϕ_j, ψ_j (prior parameter τ_{jk} in Eq. (5.30)).

The sample mean and variance (over the mixtures) of the mixture means μ_{jk} for state j of the ML estimate are reasonable estimates for the parameters ω_{jl} and ς_{jl}, i.e. ω_{jl} and ς_{jl} are chosen to be

$$\omega_j = \frac{1}{M}\sum_{k=1}^{M}\mu_{jk} \tag{5.32}$$

$$\varsigma_j = \frac{1}{M}\sum_{k=1}^{M}(\mu_{jk}-\omega_j)^2. \tag{5.33}$$

Likewise, the sample mean and variance of the mixture variances for state j of the ML estimate are reasonable estimates for the mean and variance of distribution $\mathcal{G}(\phi_{jl}, \psi_{jl})$. Letting the sample mean of the mixture variance be $\bar{\sigma}_j = \frac{1}{M}\sum_{k=1}^{M}\sigma_{jk}$, and using Eqs. (A.14) and (A.15), the parameters ϕ_j and ψ_j are therefore estimated as follows

$$\phi_j = \frac{\bar{\sigma}_j^2}{\frac{1}{M}\sum_{k=1}^{M}(\sigma_{jk} - \bar{\sigma}_j)^2} \tag{5.34}$$

$$\psi_j = \frac{\bar{\sigma}_j}{\frac{1}{M}\sum_{k=1}^{M}(\sigma_{jk} - \bar{\sigma}_j)^2}. \tag{5.35}$$

The hyperparameters C_m and C_v (defined in Eq. (5.31)) are determined by the user and express our trust in the ML estimate. Large values of C_m will result in prior distributions which are peaked around the mode of the posterior, while small values of C_m will result in a relatively non-informative prior distribution. The effect of these hyperparameters on the performance of the system will be determined in the Section 5.4.

5.3.3 Refreshing hyperparameters

In the stochastic dynamics method (Section 5.2.2), before using the leapfrog integration scheme, the momenta are "refreshed" using Gibbs sampling. It is at this same step in the stochastic dynamics method, where we will also update or "refresh" the prior parameters using Gibbs sampling.

As a result of using hyperparameters, we have to obtain the prior using Eq. (5.10). This is accomplished by Gibbs sampling of the hyperparameters γ after each transition (i.e. before leapfrog integration) of the stochastic dynamics or hybrid Monte Carlo algorithm, which results in the numerical integration of Eq. (5.10).

The hyperparameters are easily sampled, as they are independent of other parameters and hyperparameters and since their distributions have been chosen to be normal and Gamma, standard techniques for generating normal and Gamma distributed variates can be used.

5.3.4 Implementation of stochastic dynamics method

Writing the posterior in the required form of an energy function as discussed in Section 5.2.2, we get the following "potential energy" function,

$$E(\theta) = -log[P(\theta)] - \sum_{i=1}^{n} log[P(\mathbf{O}_i|\theta)]. \tag{5.36}$$

The implementation of the stochastic dynamics method is then as follows:

1. Refresh momenta and prior parameters using Gibbs sampling.
2. Starting with the current state perform L leapfrog steps to generate a new configuration. Here we require $\frac{\partial E(t)}{\partial \theta}$ (Eqs. (5.18) and (5.20)). The energy function is the same as that used for the gradient-based MAP algorithm in Chapter 4 (Eqs. (4.47) to (4.55)), and the gradient of the energy function with respect to the HMM parameters is therefore exactly the same. The derivation thereof is not repeated here.
3. Keep current configuration, repeat steps 1 through 3 until N samples are generated.

A segmental approach is used, in which we use the best state sequence, as opposed to the sum of all possible sequences, i.e. the probability $\max_{q} P(\mathbf{O}, \mathbf{q}|\theta)$ is used as an approximation of $P(\mathbf{O}|\theta) = \sum_{\text{all } \mathbf{q}} P(\mathbf{O}, \mathbf{q}|\theta)$. An embedded version has also been implemented, which uses the modified-Viterbi trellis search (Section 2.1.5) to obtain the best HMM and state sequence.

5.3.5 Recognition

In a Bayesian framework, we evaluate Eq. (5.6) for each class and classify using the following decision rule

$$C(\mathbf{O}) = C_i \quad \text{where} \quad i = \underset{j}{argmax} P_j(\mathbf{O}_{unknown}|\mathbf{O}_1, \ldots, \mathbf{O}_n). \tag{5.37}$$

Implementation of this decision rule is straightforward when applied to discrete (or label-based) phoneme or word classification: use Eq. (5.11) to numerically determine $P_j(\mathbf{O}_{unknown}|\mathbf{O}_1, \ldots, \mathbf{O}_n)$ for each class (phoneme or word).

The implementation of Eq. (5.37) for continuous speech recognition is not a trivial task. Here, a class is a string of phonemes or words and there are many possible combinations thereof. The direct implementation of Eq. (5.11), although relatively simple to implement, would therefore not be viable in terms of computation complexity.

An alternative approximation would be to use an N-best search to generate the N_s best strings and then evaluate Eq. (5.11) for only these combinations. Although this is a reasonable approximation, it is still somewhat computationally expensive. This approximation, although considerably less computationally expensive, is at best $N_s \cdot N_m$ times slower than the equivalent standard HMM recognizer. Given that N_s will typically have to be relatively large, this technique will unfortunately not be feasible either.

The following is a simple, yet reasonable approximation of the classification process described by Eq. (5.37). Although we wish to obtain $P(\mathbf{O}_{unknown}|\mathbf{O}_1, \ldots, \mathbf{O}_n)$, it is not necessary to do it directly using Eq. (5.11). Formulating the problem in terms of a single HMM, and not the sum of several sampled HMMs, we can write $P(\mathbf{O}_{unknown}|\mathbf{O}_1, \ldots, \mathbf{O}_n)$ as follows:

$$P(\mathbf{O}_{unknown}|\mathbf{O}_1,\ldots,\mathbf{O}_n) = \sum_{\text{all } \mathbf{q}} P(\mathbf{O}|\mathbf{q},\mathbf{O}_1,\ldots,\mathbf{O}_n)P(\mathbf{q}|\mathbf{O}_1,\ldots,\mathbf{O}_n), \tag{5.38}$$

where $\mathbf{q}$ is a state sequence $(q_1 q_2 \ldots q_T)$. The probability of a new observation $\mathbf{O}$ given the state sequence is

$$P(\mathbf{O}|\mathbf{q},\mathbf{O}_1,\ldots,\mathbf{O}_n) = \prod_{t=1}^{T} P(\mathbf{o}_t|q_t,\mathbf{O}_1,\ldots,\mathbf{O}_n), \tag{5.39}$$

where independence of observations has been assumed. The probability of a single observation $\mathbf{o}_t$ given a state sequence can be written as

$$P(\mathbf{o}_t|q_t,\mathbf{O}_1,\ldots,\mathbf{O}_n) = \int P(\mathbf{o}_t|q_t,\theta)P(\theta|\mathbf{O}_1,\ldots,\mathbf{O}_n)d\theta, \tag{5.40}$$

where we have integrated with respect to the posterior of the Gaussian mixture parameters. We now numerically integrate Eq. (5.40) using N_m samples of the posterior distribution $P(\mathbf{o}_t|\theta^{(m)})$,

$$\begin{aligned} P(\mathbf{o}_t|q_t,\mathbf{O}_1,\ldots,\mathbf{O}_n) &\approx \sum_{m=1}^{N_m} P(\mathbf{o}_t|\theta^{(m)}) \\ &= \sum_{m=1}^{N_m} b_{q_t}^{(m)}(\mathbf{o}_t), \end{aligned} \tag{5.41}$$

where $b_{q_t}^{(m)}(\mathbf{o}_t)$ is the observation probability of $\mathbf{o}_t$ for state q_t and the mth sample of the posterior distribution.

It remains to decide on $P(\mathbf{q}|\mathbf{O}_1,\ldots,\mathbf{O}_n)$ in Eq. (5.38). Note that we have not allowed the posterior of the transition probabilities to be sampled (as discussed in Section 5.3.1) – a reasonable assumption would be that the posterior of the transition probabilities is reasonably peaked and the ML estimate is a reasonable approximation, therefore

$$P(\mathbf{q}|\mathbf{O}_1,\ldots,\mathbf{O}_n) \approx P(\mathbf{q}|\theta) = \pi_{q_0}\prod_{t=1}^{T} a_{q_t q_{t+1}}. \tag{5.42}$$

The probability of the observation sequence $\mathbf{O}_{unknown}$ given the training data $\mathbf{O}_1,\ldots,\mathbf{O}_n$ for a given state sequence $\mathbf{q}$ can therefore be written as

$$P(\mathbf{O}_{unknown}|\mathbf{q},\mathbf{O}_1,\ldots,\mathbf{O}_n) \approx \pi_{q_0}\prod_{t=1}^{T}\left[(\sum_{m=1}^{N_m} b_{q_t}^{(m)}(\mathbf{o}_t))a_{q_t q_{t+1}}\right]. \tag{5.43}$$

Therefore, the procedure described above results in integration with respect to the posterior of the Gaussian mixtures each time an observation probability is required. This is, as opposed to integrating with respect to the full HMM posterior as in Eq. (5.6).

Viterbi implementation Here, where we would have used a single observation probability in the normal Viterbi algorithm, we now calculate the sum of the observation probabilities for the posterior samples (Eq. (5.41)). The implementation of the Viterbi algorithm for this approximation is therefore almost trivial. To find the single best state sequence, $\mathbf{q} = (q_1 q_2 \ldots q_T)$, for the given observation sequence $\mathbf{O} = (\mathbf{o}_1\mathbf{o}_2\ldots\mathbf{o}_T)$, we use the following modified Viterbi algorithm:

1. Initialization

$$\delta_1(i) = \pi_i(\sum_{m=1}^{N_m} b_i^{(m)}) \quad 1 \leq i \leq N \tag{5.44}$$

$$\psi_1(i) = 0 \tag{5.45}$$

2. Recursion

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] \sum_{m=1}^{N_m} b_j^{(m)} \tag{5.46}$$

$$\psi_t(j) = arg \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] \tag{5.47}$$

3. Termination

$$P^* = \max_{1 \leq i \leq N} [\delta_T(i)] \tag{5.48}$$

$$q_R^* = arg \max_{1 \leq i \leq N} [\delta_T(i)] \tag{5.49}$$

4. Path backtracking

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, \ldots, 1. \tag{5.50}$$

Extension to the trellis search, which is a frame-synchronous, modified implementation of the Viterbi search, is trivial. This approximation has the advantage over direct implementation of Eq. (5.11), and the N-best approximation, that it is only approximately N_m times slower than a standard HMM recognizer, where N_m is the size of the sample.

5.4 Experiments

The goal of this section is to experimentally determine the utility of the Bayesian learning approach described in this chapter. The hypothesis that will be tested here, is that "Bayesian learning will improve performance markedly in situations where little data is available for training purposes". In situations where sufficient training data is available, the posterior will be peaked sharply about the MAP point and there will

therefore be little advantage in using Bayesian learning rather than MAP estimation under such conditions.

The three datasets SUNSpeech, TIMIT and TIDIGITS are once again used here. However, the assumption here is that there is only a small amount of data available with no non-task-specific data available for adaptation purposes. The results obtained using Bayesian learning will be compared with that obtained using the adaptation techniques described in Chapter 4 where non-task-specific data was assumed to be available.

Unless otherwise stated, the stochastic dynamics method was run for 100 iterations, i.e. 100 samples were generated. The last N_m samples were used for recognition tasks. For example, a recognition experiment using $N_m = 30$ will therefore use the last 30 samples, i.e. samples 71 through to 100.

Bayesian learning is not a deterministic process and there will therefore be a certain variance in performance for the resultant systems. Each experiment was therefore repeated 10 times, so as to provide an indication of the variance in accuracy that results. Error bars are given for each result, indicating the minimum, mean and maximum accuracy for a given configuration.

5.4.1 SUNSpeech

This section evaluates the Bayesian learning algorithm under the same conditions as that used to test the adaptation techniques in Section 4.6.1, except that here, there is no English training data available for cross-language adaptation purposes. The effect of the sample size, parameters C_m, C_v, simulation time T and leapfrog step size L will be experimentally determined for the SUNSpeech dataset in this section. Results will be presented for both the small (A_S) and full (A) Afrikaans training sets. The speaker independent Afrikaans testing set is used for evaluation purposes.

Table 5.2 presents the Afrikaans test set accuracy for the two training sets available,

namely the full Afrikaans training set A in the SUNSpeech dataset and the reduced training set A_S.

Table 5.2: Base system accuracy for SUNSpeech Afrikaans test set

Training set	Mixtures	
	5	10
Afrikaans train (A)	48.6	51.5
Afrikaans adapt (A_S)	42.5	41.2

The following results are for a 3 state, 5 mixture HMM system, when only the small Afrikaans training set is available. Selected results for the full dataset, as well as 3 state, 10 mixture HMMs will be presented later in this section.

Effect of sample size Figure 5.3 shows the effect of the sample size N_m on the performance of the system. Firstly, it is noticeable that the performance for a single sample, is already better than that attained by the ML estimate. This improvement is primarily due to the regularization effect of the chosen prior and associated hyperparameters. The choice of prior and hyperparameters described in Section 5.3.2 therefore seems justified.

Considerable improvements result from increasing the sample size, with 46.1% being attained when using a sample size of 90 (a relative improvement in error rate of 6.2%). The sudden decrease in recognition rate for 100 samples, is indicative of the fact that the algorithm had not converged within the first 10 transitions (samples) of the stochastic dynamics method.

The variance on the resulting systems is relatively high, but as expected decreases as the number of samples used is increased. We would expect the variance in the solution to tend towards zero as the number of samples tended towards infinity, i.e. the numerical integration becomes the exact integration in Eq. (5.6).

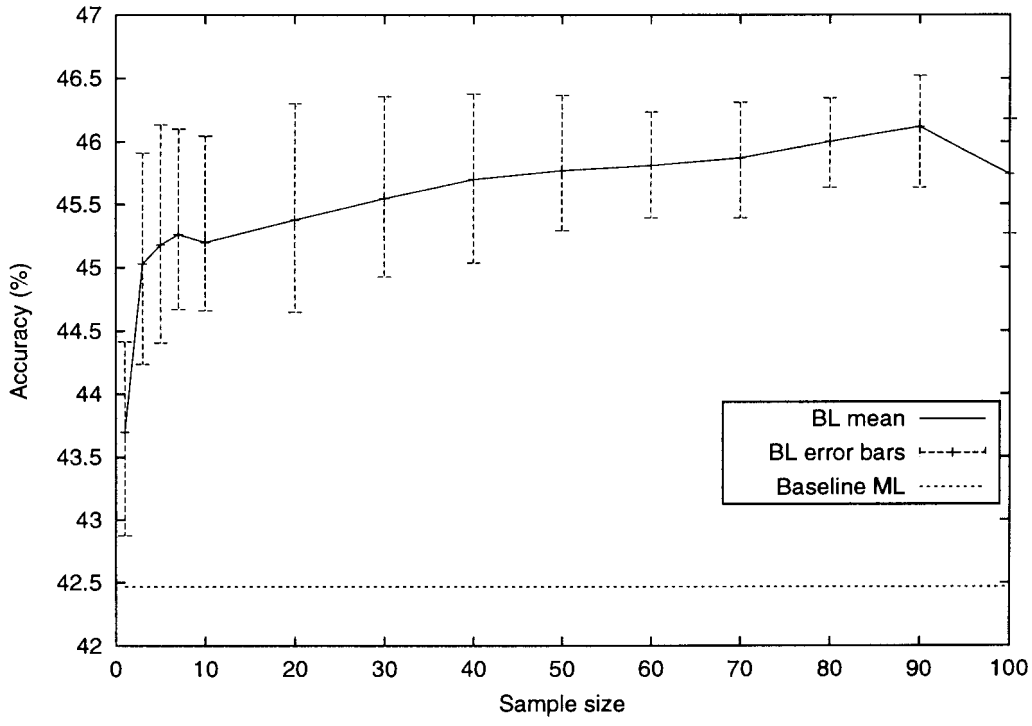


Figure 5.3: Performance of the Bayesian system (BL) versus the size of the sample used to numerically integrate with respect to the posterior.

Effect of the hyperparameter C_m As mentioned in Section 5.3.2 the parameter C_m, along with C_v, indirectly determines the degree to which the prior is peaked about its mode. It contains our *a priori* trust in the ML model used as starting point.

Figure 5.4 presents the effects of C_m on the performance of a 3 state, 5 mixture HMM when the reduced training set is used. As can be seen, an optimal value for C_m exists. Values of C_m which are too low do not force the regularization of the prior to be of any consequence. Alternatively, values which are too large result in a prior which is too restrictive and therefore unnecessarily restricting the posterior.

Figure 5.5 shows the results for a 3 state, 10 mixture system versus the parameter C_m. Note that, here the optimal value of C_m is lower ($C_m = 5$ or $C_m = 10$) than that obtained for the 3 state, 5 mixture system. The ML estimate for the more complex 10 mixture HMM system, is worse when little data is available (Table 5.2). The ML estimate for the 10 mixture system can therefore not be "trusted" as much as the 5

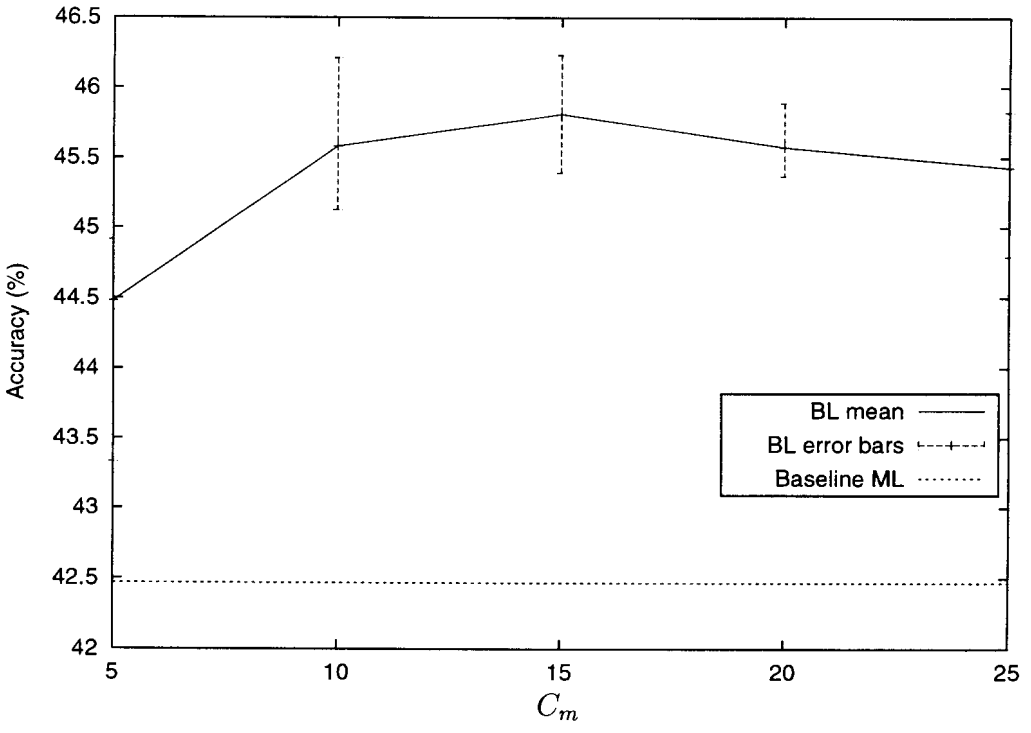


Figure 5.4: Performance of the Bayesian system (BL) versus the hyperparameter C_m, for $C_v = 4$, $L = 100$ and using a sample size of 60.

mixture system. A smaller value of C_m which makes the prior less informative, therefore results in the peak performance for this system.

Effect of the hyperparameter C_v Figure 5.6 presents the effects of the C_v on the performance of a 3 state, 5 mixture HMM when the reduced training set is used. Again, extreme values of this parameter result in degradation in performance. Fortunately, the performance of the system is relatively invariant to reasonable variations in this parameter.

Simulation time T A detailed discussion of the effect of the simulation time T was given in Section 5.2.2, and is critical to the optimal usage of the stochastic dynamics method. We therefore need to experimentally determine the effect of the number of leapfrog steps used on the performance of the system. The step size is kept constant

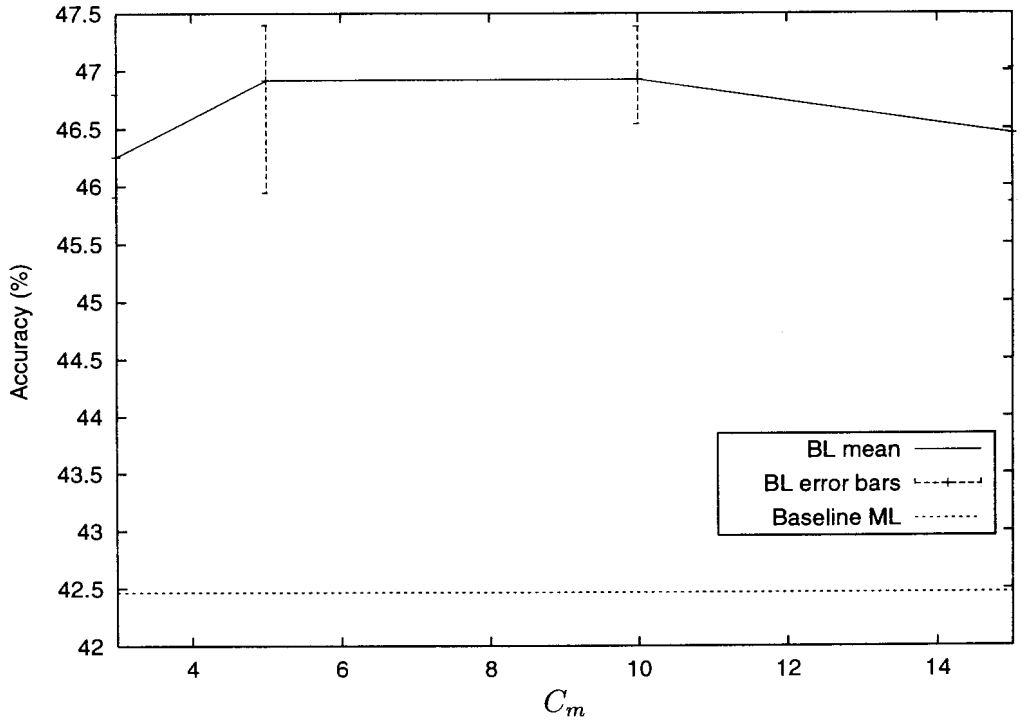


Figure 5.5: Performance of the Bayesian system (BL) versus the hyperparameter C_m, for a 3 state, 10 mixture HMM and $C_v = 4$, $L = 100$ and using a sample size of 60.

for this experiment, at $\epsilon = 0.001$.

Figure 5.7 shows the performance of the Bayesian system versus the number of leapfrog steps L, for the configuration $C_m = 15$, $C_v = 4$ and using a sample size $N_s = 60$. Remember that the simulation time is approximately equal to the step size ϵ multiplied by the number of leapfrog steps, i.e. $T \approx \epsilon L$. We expect that too few leapfrog steps result in samples which are highly correlated, which results in a sample which is not representative of the posterior. This can be seen in Figure 5.7, where using 10 leapfrog steps results in relatively poor system performance, with the minimum accuracy being less than that of the ML baseline system.

System accuracy increases as the number of leapfrog steps increase, and peaks at $L = 100$, where an average accuracy of 45.8% is attained. However, as mentioned in Section 5.2.2, values of L which are too large will not only waste time, but could worsen performance due to higher correlation between samples, resulting from the

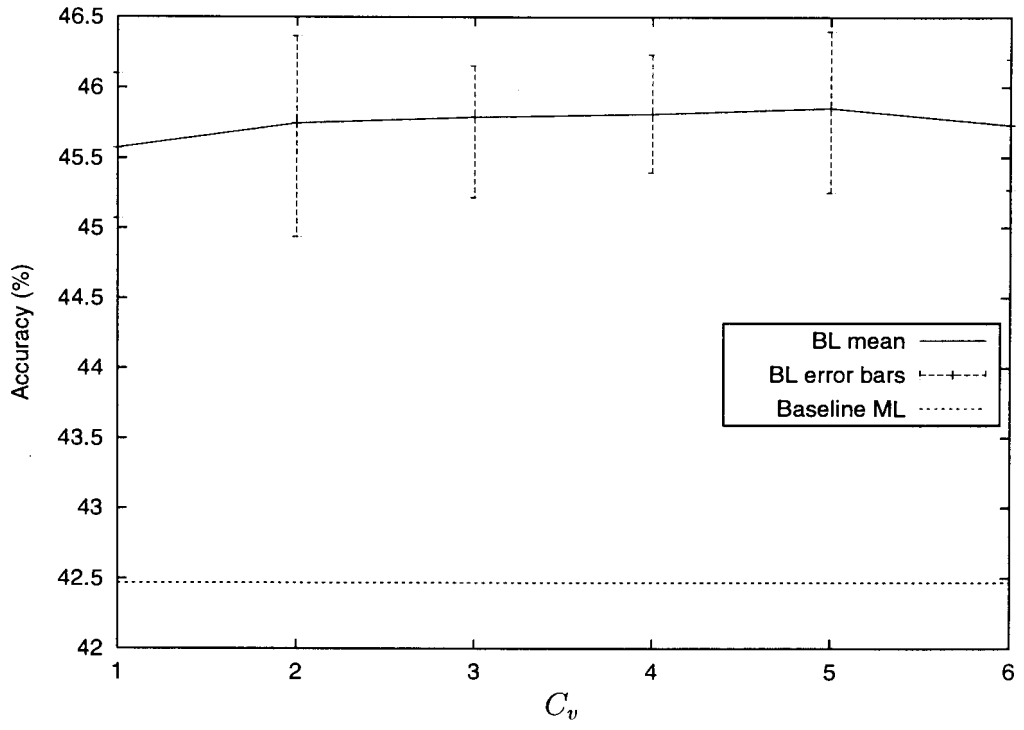


Figure 5.6: Performance of the Bayesian system (BL) versus the hyperparameter C_v, for $C_m = 15$, $L = 100$ and using a sample size of 60.

periodic nature of Hamilton's equations. This effect is also observed in Figure 5.7 where accuracy at $L = 200$ (45.75%) is slightly worse than that at $L = 100$.

Leapfrog step size Table 5.3 presents the results for the Bayesian system using step sizes of $\epsilon = 0.001$ and $\epsilon = 0.0002$. The results presented until now have used a learning rate of $\epsilon = 0.0001$. The number of steps used have been adjusted such that the simulation time T is approximately the same. This ensures that the simulation time T does not affect the results, as seen in the previous experiment. Using a step size of $\epsilon = 0.0002$ results in an average absolute improvement of approximately 0.3%. Using the smaller step size $\epsilon = 0.0002$, however, requires that 500 leapfrog step be used (for equivalent simulation time). The extra 0.3% in absolute performance is, however, not enough to justify the increase in simulation time (5 times).

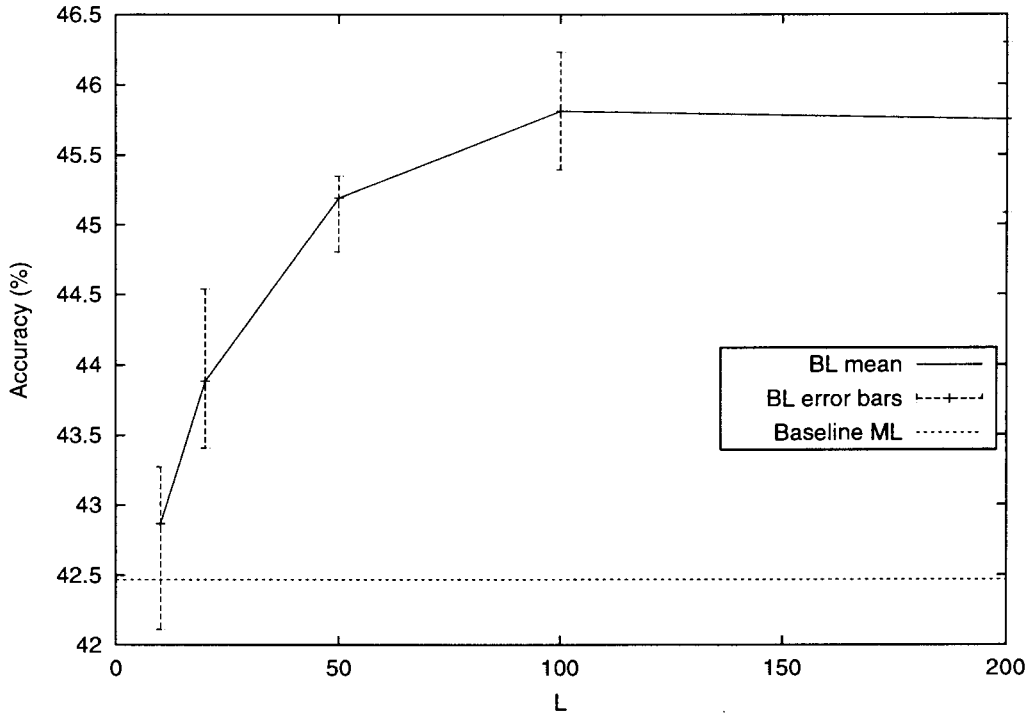


Figure 5.7: Performance of the Bayesian system (BL) versus the number of leapfrog steps L per transition, for $C_m = 15$, $C_v = 4$ and using a sample size of 60.

Summary of results Table 5.4 gives the best results obtained for the SUNSpeech dataset when using the Bayesian learning algorithm described in this chapter. Note that here, the 10 mixture model always performs best. This is as opposed to the ML estimate, where the 10 mixture system performed worse that the 5 mixture system when the small training set is used (Table 5.2).

Peak performance of 47.0% accuracy was attained using 3 state, 10 mixture HMMs and

Table 5.3: Comparison of test set accuracy for leapfrog integration using differing step sizes ($\epsilon = 0.001$ and $\epsilon = 0.0002$) but having equivalent simulation time $T \approx \epsilon L$

Leapfrog configuration	Sample size									
	10	20	30	40	50	60	70	80	90	100
$\epsilon = 0.001, L = 100$	45.2	45.4	45.5	45.7	45.8	45.8	45.9	46.0	46.1	45.7
$\epsilon = 0.0002, L = 500$	45.6	45.8	46.0	45.9	46.1	46.1	46.3	46.3	46.4	46.0

Table 5.4: Summary of the results obtained using Bayesian learning with the SUNSpeech dataset, for $L = 100$, $\epsilon = 0.001$. Relative improvement in error rate expressed as a percentage, relative to the associated baseline performance given in Table 5.2 is given in brackets.

Training set	Mixtures	Configuration	Sample size		
			10	50	90
A_S	5	$C_m = 15, C_v = 4$	45.2 (4.7)	45.8 (5.7)	46.1 (6.2)
A_S	10	$C_m = 5, C_v = 4$	**46.5** (9.0)	**46.9** (9.7)	**47.0** (9.9)
A	5	$C_m = 10, C_v = 4$	50.5 (3.7)	50.3 (3.3)	50.3 (3.3)
A	10	$C_m = 10, C_v = 4$	**53.7** (4.5)	**54.0** (5.2)	**54.1** (5.4)

a sample size of 90 for the small training set. This equates to a relative improvement in error rate of 9.9%. The same system using only 10 samples attained a 46.5% testing set accuracy (9.0% relative improvement in error rate).

When using the full training set, it is once again the 10 mixture system which performs best. Peak performance of 54.1% is attained for the 90 samples system, which is a relative improvement in error rate of 5.4%. The considerably smaller 10 sample system gives a 4.5% relative improvement in error rate (53.7% accuracy).

Whether one can justify using the 90 sample system, which is 9 times slower than the 10 sample system and approximately 90 times slower than that of the equivalent standard HMM system, is currently unlikely.

5.4.2 TIMIT

This section evaluates the Bayesian learning algorithm under the same conditions as that used to test the MCE algorithm in Section 3.4, except that here, only the small TIMIT training set T_S will be used. The procedure followed in selecting this dataset has been described in Section 2.4.1. The full TIMIT testing set has been used for all experiments presented in this section. A 3 state, 5 mixture HMM is used for all

experiments in this section. The base system test set performance for the 3 state, 5 mixture HMM when using the small TIMIT training set is 52.6%.

Here, so as not to unnecessarily duplicate results, only the effect of sample size and parameter C_m will be experimentally evaluated – to show that performance is similar for another dataset and language.

Effect of sample size Figure 5.8 shows the effect of the sample size N_m on the performance of the Bayesian learning system. The performance for a sample size of 1 ($N_m = 1$) is, once again, in general better than that of the Baseline ML system – further evidence that the choice of prior and hyperparameter is justified.

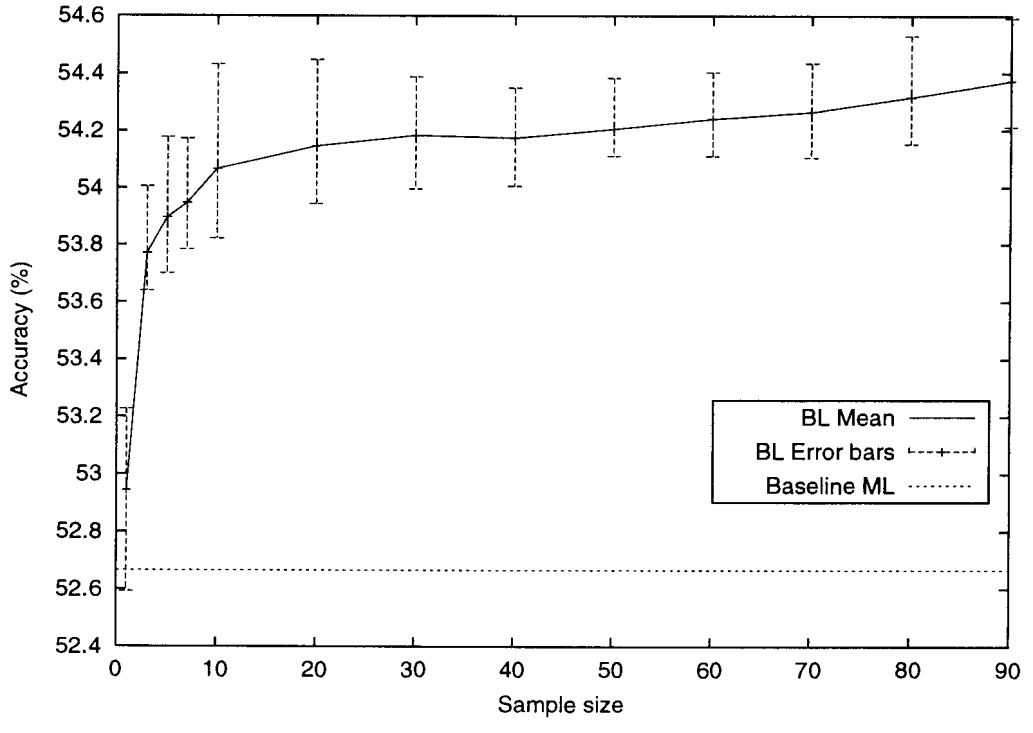


Figure 5.8: Performance of the Bayesian system (BL) versus the size of the sample used to numerically integrate with respect to the posterior.

As with the SUNSpeech experiments, by far the greatest improvements resulted when using 10 or fewer samples. Using 10 samples results in a system accuracy of 54.1%. The performance increases to a reasonable degree for sample sizes greater than 10,

though whether the extra computational effort can be justified is once again somewhat doubtful. Maximum performance of 54.4% is attained when using a sample size of 90, though better performance would probably result when using a larger sample size. As expected, the variance of the system accuracy does, in general, tend to decrease as the sample size grows.

Effect of the parameter C_m Figure 5.9 presents the effect of the parameter C_m on the performance of a 3 state, 5 mixture HMM based Bayesian system. Once again, an optimal value exists ($C_m = 15$), with the performance of the system deteriorating for values smaller or larger than this optimal value.

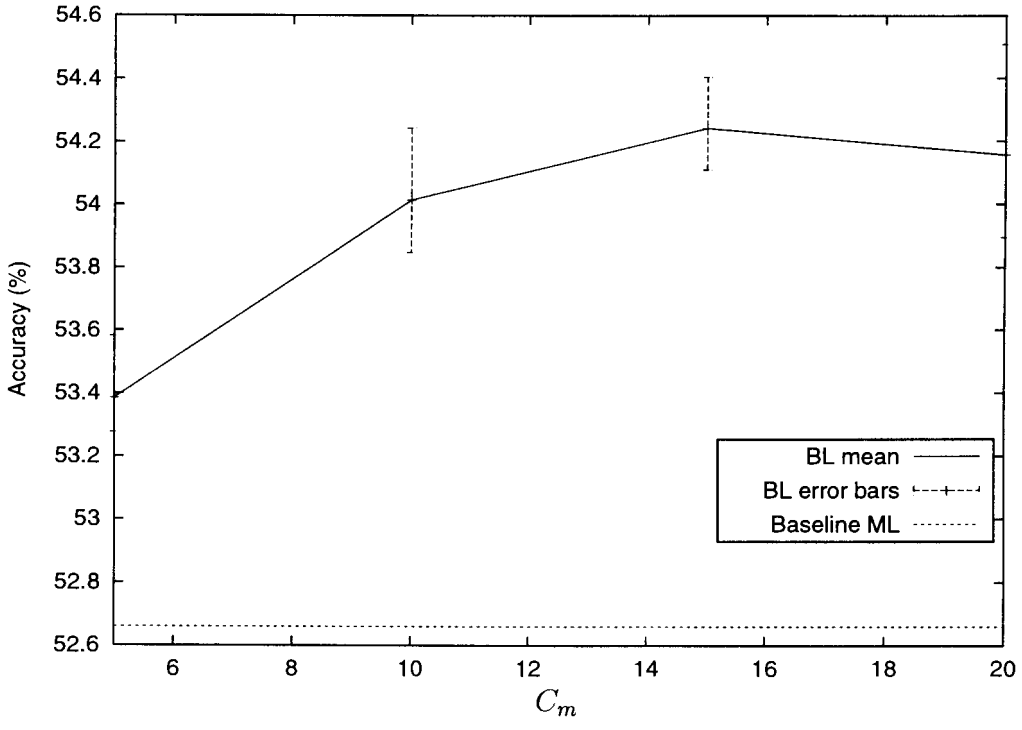


Figure 5.9: Performance of the Bayesian system (BL) versus the hyperparameter C_m, for $C_v = 4$, $L = 100$ and using a sample size of 60.

5.4.3 TIDIGITS

This section evaluates the Bayesian learning algorithm for a connected digit recognition task. The TIDIGITS database, as described in Section 2.4.2, is used for this purpose, where a small gender-independent dataset was created. The reduced speaker set (T_S) was created, using five randomly chosen female speakers and five randomly chosen male speakers, using all 77 digit sequences per speaker. The relevant training and testing sets are summarized in Table 5.5.

The full TIDIGITS testing set has been used for the experiments presented in this section. An 8 state, 5 mixture HMM is used for all experiments. The base system test set performance for the 8 state, 5 mixture HMM when using the small TIDIGITS training set is 95.8%. Once again, so as not to unnecessarily duplicate results, only the effect of sample size and parameter C_m will be evaluated.

Table 5.5: Training and testing sets used with the TIDIGIT database

Description	Label	Number of speakers			Digit sequences	Duration (minutes)
		Male	Female	Total		
Training sets:						
Full (standard)	T	55	57	112	8623	253.4
Man	T_M	55	0	55	4235	121.9
Small	T_S	5	5	10	770	21.5
Testing set		55	57	113	8623	254.4

Effect of sample size Figure 5.10 shows the performance of the Bayesian learning system versus the sample size N_m, with $C_m = 15$, $C_v = 4$ and $L = 100$. As was the case for the previous two datasets, the performance for a sample size of 1 ($N_m = 1$) resulted in better performance (mean of 96.3%) than that of the Baseline ML system (95.8%). A small improvement in accuracy (96.5%) is observed when a sample size of two is used. However, larger sample sizes result in degradation of system accuracy,

with accuracy dropping back down to 96.3% when 30 samples are used.

A connected digit task is relatively simple compared to continuous phoneme recognition. We can therefore expect that the posterior distribution for the HMM we are estimating will not be as complex as that of the two scenarios previously used. For a very simple task, one expects that a single mode posterior would result, and if the posterior is peaked no significant improvement in performance will be attained when using the Bayesian learning approach described in this chapter. With the connected digit task, however, we do observe a small, though significant, improvement in performance when using a sample size which is greater than one. The posterior of this task is therefore slightly more complex than a single mode. The improvement in performance may, for example, be attributable to a bi-modal posterior, with one mode representing male speakers and the other mode representing female speakers.

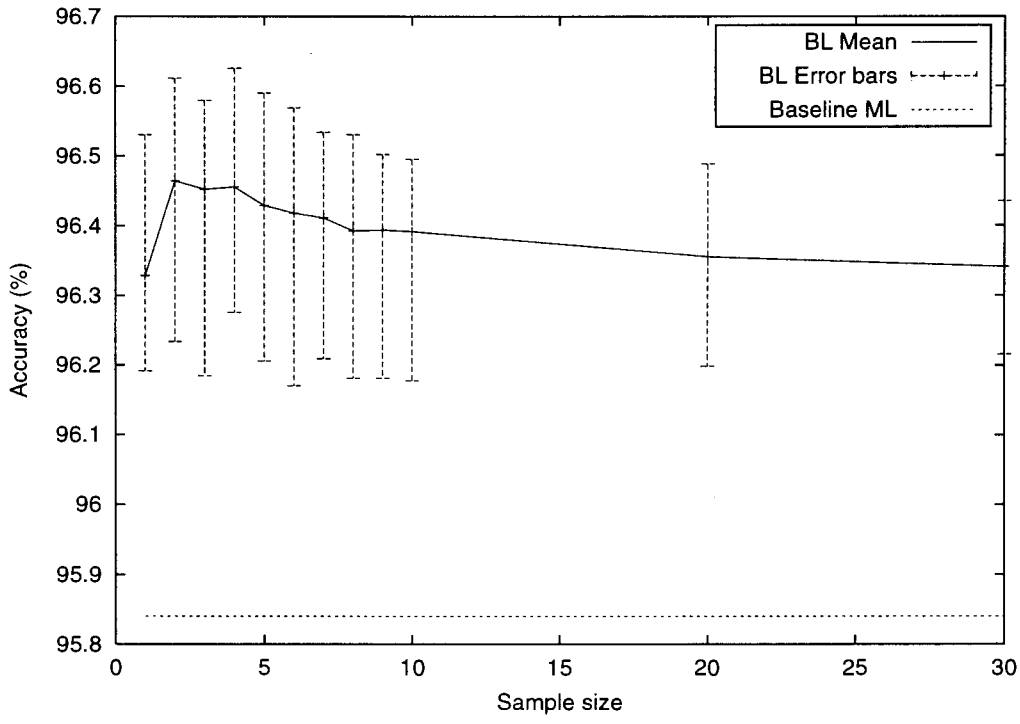


Figure 5.10: Performance of the Bayesian system (BL) versus the size of the sample used to numerically integrate with respect to the posterior.

One would, however, not expect the performance to deteriorate as the sample size

increases. A decrease in performance indicates that the distribution of the samples generated using the Markov chain Monte Carlo method is not that of the desired distribution. In Section 5.4.1 an example of this phenomenon was observed, when samples from a Markov chain were used before it had converged. Here, however, the Markov chain has converged. The decrease in accuracy observed here is probably due to the systematic errors introduced by the leapfrog algorithm. A finite step size will result in a posterior sample which is slightly different from that of the true distribution. As the sample size increases, the probability of including a sample which is not representative of the true or desired distribution therefore increases, and as a result, the performance of the system will tend to decrease as larger samples are used. This phenomenon has only been observed here, and its effect is relatively small compared to the improvements resulting when the posterior is relatively complex.

Effect of the parameter $\mathbf{C_m}$ The performance of the system (not shown graphically) is somewhat invariant to the parameter C_m, probably due the the fact that little or no improvement results from using the Bayesian learning procedure. Peak mean performances of 96.47%, 96.46% and 96.50% are attained for C_m values of 10, 15 and 20 respectively.

5.4.4 Summary of results

In Section 5.4.1 the Bayesian learning procedure introduced in this chapter was tested in a situation where a limited amount of training data was available for a new language specific recognizer. The SUNSpeech dataset was used for this purpose. The effects of the sample size, parameters C_m and C_v, simulation time T and leapfrog step size ϵ were determined for this scenario. Significant improvements in accuracy were attained when using the Bayesian learning procedure. Relative improvements in mean error rates of up to 9.9% and 5.4% were attained when using the small and full Afrikaans training sets respectively.

A small gender independent training set was created using the TIMIT dataset, which was used to test the system under much the same conditions (Section 5.4.2). A 3.8% relative improvement in mean error rate was attained for this scenario. The effect of sample size and the parameter C_v were found to be very similar to that observed when using the SUNSpeech database.

In Section 5.4.3 the system was tested using a connected digit recognition task. The TIDIGITS database was used to create a small gender-independent dataset, so as to simulate the scenario where little training data was available for training a connected digit recognizer. A relative improvement in error rate of 16.7% was attained when using a sample size of two. Using only one sample resulted in an 11.9% relative improvement, due to the choice of prior and hyperparameters. A decrease in performance was observed for sample sizes greater than two, which was ascribed to the systematic errors resulting from the leapfrog integration algorithm.

If one was to have the luxury of having an independent cross-validation set, then the resultant performances can be expected to be even better. Under these conditions, one could create several samples of the posterior and then choose the best sample using the cross-validation set. Although this will not necessarily give the truly best sample, it will probably improve results such that they are closer to the maximum results attained in this section.

5.5 Summary

This chapter introduced Bayesian learning and its usage within a continuous speech recognition framework. Markov chain Monte Carlo methods, which are used to sample the posterior, were described. Implementation specifics such as maintaining the HMM constraints, choice of the prior distributions and parameters, and an efficient implementation for recognition were also discussed.

Section 5.4 experimentally evaluated the Bayesian learning procedure introduced in this chapter. The SUNSpeech, TIMIT and TIDIGITS databases were used for this purpose. The effects of sample size, hyperparameters C_m and C_v, simulation time T and leapfrog step size ϵ were determined. Significant improvements in system accuracy were attained for the scenarios created using all three datasets. The improvements realized for the connected digit task (TIDIGITS) were, however, mainly due to the regularization effect of the prior, with a sample size of two resulting in peak performance. This was attributed to the fact that the connected digit task is relatively simple and the posterior thereof is therefore probably also relatively simple.

The results attained show that there is much promise in the usage of the Bayesian learning procedure proposed. Although considerably more computationally expensive than most HMM training techniques, I believe, however, that the improved performance warrants the additional computational expense.

Chapter 6

Conclusion

6.1 Overview

The problem addressed in this thesis is the design of a speech recognizer when only sparse training data are available. The recording and subsequent segmenting and annotation of large speech databases is an expensive and labour-intensive process. The aim of the techniques developed in this thesis is therefore to improve recognition performance of hidden Markov model (HMM) systems when training data is limited.

This thesis started by considering the minimum classification error (MCE) parameter estimation procedure. The MCE criterion is closely related to the goal of ideal Bayes classification. Overspecialization is, however, prevalent when using MCE, especially when training data is limited. Several modifications to the standard MCE procedure were therefore proposed which limited the effect of overtraining. The MCE algorithm was used within a training framework (Section 1.2).

The usage of the maximum *a-posteriori* (MAP) estimation algorithm was investigated for situations where little task-specific training data is available, but a reasonable quantity of non-task-specific training data can be used. The MAP estimation procedure

was used within an adaptation framework (as described in Section 1.1) in this thesis, where non-task-specific data was used in conjunction with limited task-specific data to improve recognition performance for a specific task.

Finally, Bayesian learning was investigated and an implementation for HMM speech recognition was proposed and implemented. Here no assumption is made about the availability of non-task-specific data, and only the task specific data is used in the training process. The Bayesian learning procedure as proposed and implemented in this work is a training algorithm (Section 1.2), where only task-specific data is used.

The algorithms proposed and tested in this thesis have been found to be extremely effective in situations where little training data is available. Significant improvements in recognition performance have been attained for the sparse data scenarios used for evaluation purposes. The algorithms proposed are, in general, considerably more computationally expensive, with execution times for the GMAP and MAPMCE being of the order of 3 hours and 10 hours respectively (depending on the task and computer used) as compared to the 13 minutes required by MAP (or ML). The Bayesian learning procedure is extremely computationally expensive with the execution time being around one to three days, depending on the task and computer used. The training of speech recognition systems is typically done offline and the time taken to estimate the model parameters is therefore often not critical. The additional computational expense can therefore be justified by the significantly improved recognition performance resulting from the use of these algorithms.

The contributions of this thesis to the field of speech recognition are,

1. New modifications to the MCE algorithm are proposed in Chapter 3. These modifications limit the effect of overtraining which is prevalent when using MCE training.

2. A new gradient-based MAP adaptation algorithm (GMAP) that does not make any assumptions concerning the form of the prior distribution was proposed in

Chapter 4. This algorithm was shown to outperform the standard MAP approach of Gauvain and Lee [45] for the conditions tested.

3. A new MCE based MAP adaptation algorithm was proposed and tested in Chapter 4. This algorithm too was shown to work better than the standard MAP approach, as well as being better than standard MCE.

4. Bayesian learning was introduced. An original implementation of Bayesian learning for hidden Markov model speech recognition was introduced and discussed. This is, to the author's knowledge, the first time that Bayesian learning using Markov chain Monte Carlo methods has been used for hidden Markov model speech recognition.

6.2 Summary by Chapter

This thesis centered on the following topics:

In *Chapter 1* the topic of speech recognition was introduced, along with the topic of data sufficiency and the effects thereof. The algorithms currently used to alleviate the problem of insufficient data were briefly reviewed.

In *Chapter 2* the basic theory of hidden Markov models and the speech recognition system used in this thesis were described. Overtraining was discussed in terms of the bias/variance dilemma. The experimental procedure common to all experiments conducted in this thesis was described and details of the three speech databases used for experimental evaluation were given.

In *Chapter 3* the minimum classification error criterion was introduced and its usage within a continuous speech recognition framework was discussed. The MCE criterion realizes optimal decision boundaries in a Bayes sense. The MCE criterion focuses on minimizing the probability of error. Classifiers trained using the MCE criterion

are therefore able to attain the goal of Bayes classification even when the modeling assumptions are incorrect. Embedded or string-level MCE, where the recognition error for entire strings is minimized, was discussed.

The effect of a smoothed zero-one loss function was discussed and experimentally determined for the TIMIT dataset. Furthermore, the need for a zero-one loss function was questioned and the conclusion was reached that there is little evidence to suggest that there is an advantage or disadvantage to using a smoothed zero-one loss function. This result was used later in modifying the MCE criterion, where the modification could not be mathematically justified when a non-linear loss function was used.

Overtraining within the MCE framework was discussed and three modifications were proposed. The first modification stops the mixture variances from becoming very small, which we expect to happen when little data is available. The second added a weighted likelihood term to the MCE criterion, thereby reinforcing correct substrings, as well as improving discrimination for incorrect substrings. Lastly, a word-based string-level MCE algorithm was proposed, in which smaller word-based substrings were used, instead of the the entire string. Significant gains in performance resulted when using these modifications with the TIMIT database. Improvements of up to 10% in relative error rate on the test set were achieved for the TIMIT dataset.

In *Chapter 4* Bayesian adaption and its usage within a continuous speech recognition framework was introduced. The MAP algorithm of Gauvain and Lee [45] was described. A gradient-based MAP algorithm (GMAP) which makes no assumption about the form of the prior was proposed and the implementation thereof was discussed. A Bayesian inspired MCE-based adaptation algorithm (MAPMCE) was also proposed. The MAPMCE algorithm is a simple extension of the MCE algorithm and its implementation is relatively simple.

The three algorithms were experimentally evaluated using the SUNSpeech database for language adaptation, and using the TIMIT and TIDIGIT databases for gender

adaptation experiments. Figure 6.1 shows the best relative improvements in error rate obtained using the MAP, GMAP and MAPMCE algorithms for the SUNSpeech, TIMIT and TIDIGITS datasets. Relative improvements in error rate of up to 10.2% were attained for the SUNSpeech dataset (using the GMAP algorithm). For the gender adaptation task using the TIMIT dataset relative improvements in error rates of up to 25.5% were attained (using the MAPMCE algorithm). Finally, relative improvements in error rate of up to 66.0% (MAPMCE) were reported for the TIDIGITS based gender adaptation task. When using duration modeling with the connected digit task (TIDIGITS), the GMAP algorithm performed best (relative improvements in error rate of 54.9%). The MAP algorithm, in general, performed somewhat worse.

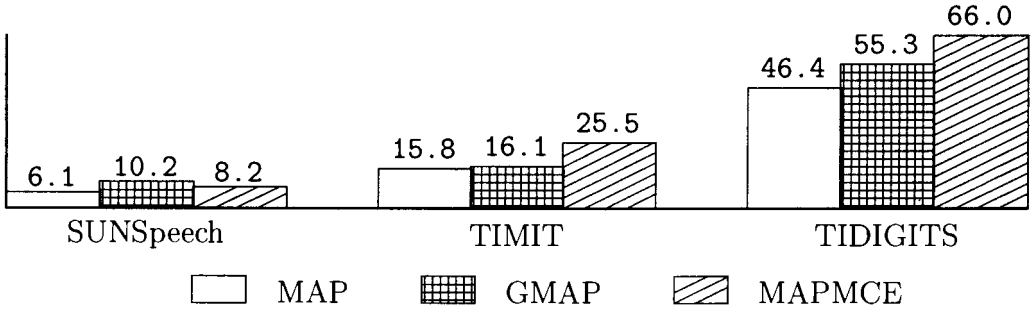


Figure 6.1: Comparison of best relative improvements in error rates obtained using the MAP, GMAP and MAPMCE algorithms for the three datasets used (SUNSpeech, TIMIT and TIDIGITS).

In *Chapter 5* Bayesian learning and its usage within a continuous speech recognition framework was discussed. Markov chain Monte Carlo methods, which are used to sample the posterior, were described. Implementation specifics such as maintaining the HMM constraints, the prior distributions and an efficient implementation for recognition were also discussed.

The SUNSpeech, TIMIT and TIDIGITS databases were used to experimentally evaluate the proposed Bayesian learning procedure. The effects of various algorithm configurations were determined. Significant improvements in system accuracy were attained for the scenarios created using the three datasets. Relative improvements in error rates of up to 9.9%, 3.8% and 16.7% were attained for the SUNSpeech, TIMIT

and TIDIGITS databases respectively. The results attained show that there is much promise in the usage of the Bayesian learning procedure proposed. Although considerably more computationally expensive than standard HMM training, the improved performance warrants the additional computational expense for certain situations.

6.3 Future research

The research performed in this thesis focused on the sparse data problem. The algorithms described were applied to a relatively limited set of tasks and languages. Future research could therefore incorporate a wider variety of typical scenarios where training data is limited, such as cross-database adaptation, speaker adaptation and dialect adaptation. The application of the techniques described here, to context dependent modeling, i.e. bi-phone or tri-phone modeling, will be an important extension of this work. A detailed comparison of Bayesian adaptation techniques with transformation based adaptation algorithms (MLLR, MAPLR) for a wide variety of sparse data scenarios will also be of much interest.

Bayesian methods can also be used for purposes other than the sparse data problem. These include using Bayesian methods to choose the optimal model configuration for a point estimate paradigm, as done by MacKay [72] in the field of neural networks.

The Bayesian learning procedure proposed can be improved upon and several aspects thereof must therefore still be investigated, they include:

- sampling the complete posterior of all HMM parameters including transition probabilities, mixture weights and if applicable, duration modeling parameters,
- using the Hybrid Monte Carlo method to sample the posterior distribution,
- making the modified Viterbi search (Section 5.3.5) more efficient, and

- other prior and hyperparameter configurations may result in even better system performance.

Bibliography

[1] V. Abrash, A. Sankar, H. Franco, and M. Cohen. Acoustic adaptation using nonlinear transformations of HMM parameters. In *Proc. ICASSP '96*, pages 729 – 732, Atlanta, GA, May 1996.

[2] S. Amari. A theory of adaptive pattern classifiers. *IEEE Trans. Electron. Comput.*, 16:299–307, June 1967.

[3] H.C. Andersen. Molecular dynamics simulations at constant pressure and/or temperature. *Journal of Chemical Physics*, 72:2384–2393, 1980.

[4] ARPA. The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus. Training and Test Data. NIST Speech Disc CD1-1.1, Dec. 1990.

[5] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occuring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, 41(1):164–171, 1970.

[6] J. R. Bellegarda, P. V. de Souza, D. Nahamoo, M. Padmanabhan, M. A. Picheny, and L. R. Bahl. Experiments using data augmentation for speaker adaptation. In *Proc. ICASSP '95*, pages 692 – 695, Detroit, MI, May 1995.

[7] J.M. Bernardo and A.F.M. Smith. *Bayesian theory.* Wiley, England, 1994.

[8] A. Biem and S. Katagiri. Feature extraction based on minimum classification error/generalized probabilistic descent method. In *Proc. ICASSP '93*, pages II–275 – II–278, Minneapolis, MN, April 1993.

[9] A. Biem and S. Katagiri. Filter bank design based on discriminative feature extraction. In *Proc. ICASSP '94*, pages I–485 – I–488, Adelaide, Australia, April 1994.

[10] J. Billa, K. Ma, J.W. McDonough, G. Zavaliagkos, D.R. Miller, K.N. Ross, and A. El-Jaroudi. Multilingual speech recognition: The 1996 Byblos callhome system. In *Proc. Eurospeech '97*, pages 363–366, Rhodes, Greece, September 1997.

[11] C.M. Bishop. Curvature-driven smoothing: A learning algorithm for feed-forward networks. *IEEE Trans. Neural Networks*, 4(5):882–884, 1993.

[12] C.M. Bishop. *Neural networks for pattern recognition.* Oxford Univ. Press, Oxford, 1995.

[13] G.E.P. Box and G.C. Tiao. *Bayesian Inference in Statistical Analysis.* John Wiley and Sons, New York, 1973,1992.

[14] P.F. Brown. *The Acoustic-Modeling Problem in Automatic Speech Recognition.* PhD thesis, Department of Computer Science, Carnegie Mellon University, 1987.

[15] U. Bub, J. Köhler, and B. Imperl. In-service adaptation of multilingual hidden-Markov-models. In *Proc. ICASSP '97*, pages 1451 – 1454, Munich, Germany, April 1997.

[16] D. Burshtein. Robust parametric modeling of durations in hidden Markov models. In *Proc. ICASSP '95*, pages 548 – 551, Detroit, MI, May 1995.

[17] P.-C. Chang and B.-H. Juang. Discriminative template training for dynamic programming speech recognition. In *Proc. ICASSP '92*, pages 493–496, San Francisco, CA, March 1992.

[18] J.-K. Chen and F. K. Soong. Discriminative training of high performance speech recognizer using N-best candidates. In *Proc. ICASSP '94*, pages I–625 – I–628, Adelaide, Australia, April 1994.

[19] J.-K. Chen and F.K. Soong. An N-best candidates-based discriminative training for speech recognition applications. *IEEE Transactions on Speech and Audio Processing*, 2(1):206–216, January 1994.

[20] J. T. Chien, H.-C. Wang, and C. H. Lee. Improved Bayesian learning of hidden Markov models for speaker adaptation. In *Proc. ICASSP '97*, pages 1027 – 1030, Munich, Germany, April 1997.

[21] J.T. Chien. In-line hierarchical transformation of hidden Markov models for speaker adaptation. In *Proc. ICSLP '98*, Sydney, Australia, November 1998.

[22] W. Chou. Maximum a posterior linear regression with elliptically symmetric matrix variate priors. In *Proc. Eurospeech '99*, pages 1–4, Budapest, Hungary, September 1999.

[23] W. Chou, B.-H. Juang, and C.-H. Lee. Segmental GPD training of HMM based speech recognizer. In *Proc. ICASSP '92*, pages 473–476, San Francisco, 1992. IEEE.

[24] W. Chou, C.-H. Lee, and B.-H. Juang. Minimum error rate training based on N-best string models. In *Proc. ICASSP '93*, pages I–652 – I–655, Minneapolis, MN, April 1993.

[25] C. Corredor-Ardoy, L. Lamel, M. Adda-Decker, and J-L. Gauvain. Multilingual phone recognition of spontaneous telephone speech. In *Proc. ICASSP '98*, Seattle, USA, May 1998.

[26] M. Creutz. Global Monte Carlo algorithms for many-fermion systems. *Physical Review D*, 38(4):1228–1238, Aug 1988.

[27] M.H. DeGroot. *Optimal Statistical Decisions*. McGraw-Hill, New York, 1970.

[28] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society series B*, 39:1–38, 1977.

[29] L. Deng. Integrated-multilingual speech recognition using universal phonological features in a functional speech production model. In *Proc. ICASSP '97*, pages 1007 – 1010, Munich, Germany, April 1997.

[30] V. Diakoloukas, V. Digalakis, L. Neumeyer, and J. Kaja. Development of dialect-specific speech recognizers using adaptation methods. In *Proc. ICASSP '97*, pages 1455 – 1458, Munich, Germany, April 1997.

[31] V.D. Diakoloukas and V.V. Digalakis. Maximum-likelihood stochachastic-transformation adaptation of hidden Markov models. *IEEE Transactions on Speech and Audio Processing*, 7(2):177–187, March 1999.

[32] V. Digalakis and L. Neumeyer. Speaker adaptation using combined transformation and Bayesian methods. In *Proc. ICASSP '95*, pages 680 – 683, Detroit, MI, May 1995.

[33] V.V. Digalakis, D. Rtischev, and L.G. Neumeyer. Speaker adaptation using constrained estimation of Gaussian mixtures. *IEEE Transactions on Speech and Audio Processing*, 3(5):357–366, September 1995.

[34] P.M. Djurić and J.H. Chun. Estimation of nonstationary hidden Markov models by MCMC sampling. In *Proc. ICASSP '99*, Phoenix, Arizona, USA, May 1999.

[35] J.A. Du Preez. Modelling durations in hidden Markov models with application to word spotting. In *Proc. IEEE South African symposium on Communications and Signal Processing*, pages 1–5, Fourways, South Africa, Aug. 1991.

[36] S. Duane. Stochastic quantization versus the microcanonical ensemble: getting the best of both worlds. *Nuclear Physics*, B257:652–662, 1985.

[37] S. Duane, A.D. Kennedy, B.J. Pendleton, and D. Roweth. Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222, Sep 1987.

[38] S. Duane and J.B. Kogut. Hybrid stochastic differential equations applied to quantum chromodynamics. *Physical Review Letters*, 55(25):2774–2777, Dec 1985.

[39] S. Duane and J.B. Kogut. The theory of hybrid stochastic algorithms. *Nuclear Physics*, B275:398–420, 1986.

[40] R.O. Duda and P.E. Hart. *Pattern classification and scene analysis.* Wiley, New York, 1973.

[41] V. Fischer, Y. Gao, and E. Janke. Speaker-independent upfront dialect adaptation in a large vocabulary continuous speech recognizer. In *Proc. ICASSP '98*, Seattle, USA, May 1998.

[42] H. Franco and A. Serralheiro. Training HMMs using a minimum recognition approach. In *Proc. ICASSP '91*, pages 357–360, Toronto, Canada, May 1991.

[43] J-L. Gauvain and C.H. Lee. Bayesian learning for hidden Markov models with Gaussian mixture state observation densities. *Speech Communication*, pages 205–213, June 1992.

[44] J-L. Gauvain and C.H. Lee. Improved accoustic modeling with Bayesian learning. In *Proc. ICASSP '93*, pages II–481–II–484, Minneapolis, MN, April 1993.

[45] J-L. Gauvain and C.H. Lee. Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains. *IEEE Transactions on Speech and Audio Processing*, 2(2):291–298, April 1994.

[46] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58, 1992.

[47] J. Glass, G. Flammia, D. Goodine, M. Phillips, J. Polifroni, S. Sakai, S. Seneff, and V. Zue. Multilingual spoken-language understanding in the MIT Voyager system. *Speech Communication*, 17:1–18, August 1995.

[48] S.J. Godsill and C. Andrieu. Bayesian seperation and recovery of convolutively mixed autoregressive sources. In *Proc. ICASSP '99*, Phoenix, Arizona, USA, May 1999.

[49] Y. Gong and J.-P. Haton. Stochastic trajectory modeling for speech recognition. In *Proc. ICASSP '94*, pages I–57 – I–60, Adelaide, Australia, April 1994.

[50] R. Gupta, G.W. Kilcup, and S. R. Sharpe. Tuning the hybrid Monte Carlo algorithm. *Physical Review D*, 38(4):1278–1287, Aug 1988.

[51] S. K. Gupta, F. Soong, and R. Haimi-Cohen. High-accuracy connected digit recognition for mobile applications. In *Proc. ICASSP '96*, pages 57 – 60, Atlanta, GA, May 1996.

[52] Q. Huo and C. Chan. On-line Bayes adaptation of SCHMM parameters for speech recognition. In *Proc. ICASSP '95*, pages 708 – 711, Detroit, MI, May 1995.

[53] Q. Huo, C. Chan, and C.-H. Lee. Bayesian learning of the SCHMM parameters for speech recognition. In *Proc. ICASSP '94*, pages I–221 – I–224, Adelaide, Australia, April 1994.

[54] Q. Huo, H. Jiang, and C. H. Lee. A Bayesian predictive classification approach to robust speech recognition. In *Proc. ICASSP '97*, pages 1547 – 1550, Munich, Germany, April 1997.

[55] I. Illina and Y. Gong. Elimination of trajectory folding phenomenon: Trajectory mixture and mixture stochastic trajectory model. In *Proc. ICASSP '97*, pages 1395 – 1398, Munich, Germany, April 1997.

[56] C. Jankowski, A. Kalyanswamy, S. Basson, and J. Spitz. NTIMIT: A phonetically balanced, continuous speech, telephone bandwidth speech database. In *Proc. ICASSP '90*, pages 109–112, Alburquerque, NM, April 1990.

[57] H. Jiang, K. Hirose, and Q. Huo. Robust speech recognition based on Viterbi Bayesian predictive classification. In *Proc. ICASSP '97*, pages 1551 – 1554, Munich, Germany, April 1997.

[58] H. Jiang, K. Hirose, and Q. Huo. Improving Viterbi Bayesian predictive classification via sequential Bayesian learning in robust speech recognition. In *Proc. ICASSP '98*, Seattle, USA, May 1998.

[59] B.-H. Juang, W. Chou, and C.-H. Lee. Minimum classification error rate methods for speech recognition. *IEEE Transactions on Speech and Audio Processing*, 5(3):257–265, May 1997.

[60] B.-H. Juang and S. Katagiri. Discriminative learning for minimum error classification. *IEEE Transactions on Signal Processing*, 40(12):3043–3054, December 1992.

[61] B.-H. Juang and L. R. Rabiner. The segmental K-means algorithm for estimating parameters of hidden Markov models. *IEEE Trans. Acoustics, Speech and Signal Processing*, ASSP-38(9), 1990.

[62] S. Kapadia, V. Valtchev, and S. J. Young. MMI training for continuous phoneme recognition on the TIMIT database. In *Proc. ICASSP '93*, Minneapolis, MN, April 1993.

[63] J. Köhler. Language adaptation of multilingual phone models for vocabulary independent speech recognition tasks. In *Proc. ICASSP '98*, pages 417 – 420, Seattle, USA, May 1998.

[64] T. Komori and S. Katagiri. Application of a generalized probabilistic descent method to dynamic time warping-based speech recognition. In *Proc. ICASSP '92*, pages 497–500, San Francisco, CA, March 1992.

[65] O.W. Kwon and C.K. Un. Performance of HMM-based speech recognizers with discriminative state-weights. *Speech Communication*, 19:197–205, 1996.

[66] C-H. Lee and J-L. Gauvain. Speaker adaptation based on MAP estimation of HMM parameters. In *Proc. ICASSP '93*, pages II–558–II–561, Minneapolis, MN, April 1993.

[67] C-H. Lee, C-H Lin, and B-H Juang. A study on speaker adaptation of the parameters of continuous density hidden Markov models. *IEEE Trans. Signal Processing*, 39(4):806–814, April 1991.

[68] K. F. Lee and H. W. Hon. Speaker-independent phone recognition using hidden Markov models. *IEEE Trans. Acoustics, Speech and Signal Processing*, ASSP-37(11):1641, 1989.

[69] C. J. Leggetter and P. C. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. *Computer Speech and Language*, 9(2):171–185, April 1995.

[70] R. G. Leonard. A database for speaker-independant digit recognition. In *Proc. ICASSP '84*, pages 42.11.1–42.11.4, 1984.

[71] L. A. Liporace. Maximum likelihood estimation for multivariate observations of Markov sources. *IEEE Transactions on Information Theory*, 28(5):729–734, Spetember 1982.

[72] David J.C. Mackay. *Bayesian Methods for Adaptive Models.* PhD thesis, California Institute of Technology, Pasadena, California, December 1991.

[73] J. S. Maritz and T. Lwin. *Empirical Bayes Methods.* Chapman and Hall, London, 1989.

[74] T. Matsui and S. Furui. A study of speaker adaptation based on minimum classification error training. In *Proc. Eurospeech '95*, pages 81–84, Madrid, Spain, September 1995.

[75] E. McDermott. *Discriminative Training for Speech Recognition.* PhD thesis, Waseda University, Japan, March 1997.

[76] E. McDermott and S. Katagiri. Prototype-based discriminative training for various speech units. In *Proc. ICASSP '92*, San Francisco, CA, March 1992.

[77] E. McDermott and S. Katagiri. Prototype-based MCE/GPD training for word spotting and connected word recognition. In *Proc. ICASSP '93*, pages II–291 – II–294, Minneapolis, MN, April 1993.

[78] E. McDermott and S. Katagiri. String-level MCE for continuous phoneme recognition. In *Proc. Eurospeech '97*, pages 123–126, Rhodes, Greece, September 1997.

[79] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, and A.H. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, Jun 1953.

[80] P. J. Moreno and R. M. Stern. Sources of degradation of speech recognition in telephone environments. In *Proc. ICASSP '94*, pages I–109 – I–112, Adelaide, Australia, April 1994.

[81] A. Nadas, D. Nahamoo, and M. A. Picheny. On a model-robust training method for speech recognition. *IEEE Trans. Acoustics, Speech and Signal Processing*, ASSP-36(9):1432, 1988.

[82] R.M. Neal. *Bayesian Learning for Neural Networks.* Springer, New York, 1996.

[83] C. Nieuwoudt. *Cross-language acoustic adaptation for automatic speech recognition.* PhD thesis, University of Pretoria, South Africa, April 2000.

[84] C. Nieuwoudt and E.C. Botha. Adaptation of acoustic models for multilingual recognition. In *Proc. Eurospeech '99*, pages 907–910, Budapest, Hungary, September 1999.

[85] C. Nieuwoudt and E.C. Botha. Cross-language use of accoustic information for automatic speech recognition. In *Proc. ICSLP 2000*, volume 3, pages 722–725, Beijing, China, October 2000.

[86] N. Nilsson. *Problem-Solving Methods in Artificial Intelligence.* McGraw Hill, New York, New York, 1971.

[87] Y. Normandin. Optimal splitting of HMM gaussian mixture components with MMIE training. In *Proc. ICASSP '95*, pages 449 – 452, Detroit, MI, May 1995.

[88] Y. Normandin and S.D. Morgera. An improved MMIE training algorithm for speaker-independent, small vocabulary, continuous speech recognition. In *Proc. ICASSP '91*, pages 537–540, Toronto, Canada, May 1991.

[89] D.W. Purnell and E.C. Botha. Improved performance and generalization of minimum classification error training for continuous speech recognition. In *Proc. ICSLP 2000*, volume 4, pages 165–168, Beijing, China, October 2000.

[90] L. Rabiner and B.-H. Juang. *Fundamentals of Speech Recognition.* Prentice-Hall, Englewood Cliffs, NJ, 1993.

[91] L. R. Rabiner, J. G. Wilpon, and F. K. Soong. High performance connected digit recognition using hidden Markov models. *IEEE Trans. Acoustics, Speech and Signal Processing*, ASSP-37(8):1214, 1989.

[92] L.R. Rabiner. An introduction to hidden Markov models. *IEEE Signal Processing Magazine*, 5:4–16, January 1988.

[93] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. of the IEEE*, 77:257–286, Feb 1989.

[94] L.R. Rabiner and B.H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, pages 4–16, January 1986.

[95] C. Rathinavelu. Minimum classification error linear regression (MCELR) for speaker adaptation using HMM with trend functions. In *Proc. Eurospeech '97*, pages 2343–2346, Rhodes, Greece, September 1997.

[96] C. Rathinavelu and L. Deng. HMM-based speech recognition using state-dependent linear transforms on mel-warped DFT features. In *Proc. ICASSP '96*, pages 9 – 12, Atlanta, GA, May 1996.

[97] H. Robbins. The empirical Bayes approach to statistical decision problems. *Annals of Mathematical Statistics*, 35:1–20, 1964.

[98] C.P. Robert, A. Doucet, and S.J. Godsill. Marginal MAP estimation using Markov chain Monte Carlo. In *Proc. ICASSP '99*, Phoenix, Arizona, USA, May 1999.

[99] S. Russel and P. Norvig. *Artificial Intelligence, A Modern Approach.* Prentice Hall, Upper Saddle River, New Jersey, 1995.

[100] A. Sankar. Robust HMM estimation with Gaussian merging-splitting and tied-transform HMMs. In *Proc. ICSLP '98*, volume 6, pages 2499–2502, Sydney, Australia, November 1998.

[101] R. Schlüter and W. Machery. Comparison of discriminative training criteria. In *Proc. ICASSP '98*, Seattle, USA, May 1998.

[102] T. Schultz and A. Waibel. Fast bootstrapping of LVCSR systems with multilingual phoneme sets. In *Proc. Eurospeech '97*, pages 371–374, Rhodes, Greece, September 1997.

[103] R. Schwartz and S. Austin. A comparision of several approximate algorithms for finding multiple (NBEST) sentence hypotheses. In *Proc. ICASSP '91*, pages 701–704, Toronto, Canada, May 1991.

[104] H. Shimodaira, J. Rokui, and M. Nakai. Improving the generalization performance of the MCE/GPD learning. In *Proc. ICSLP '98*, Sydney, Australia, November 1998.

[105] O. Siohan, A.E. Rosenberg, and S. Parthasarathy. Speaker identification using minimum classification error training. In *Proc. ICASSP '98*, Seattle, USA, May 1998.

[106] F. K. Soong and E.-F. Huang. A tree-trellis based fast search for finding the N best sentence hypotheses in continuous speech recognition. In *Proc. ICASSP '91*, pages 705–708, Toronto, Canada, May 1991.

[107] J. Su, H. Li, J.-P. Haton, and K.-T. Ng. Speaker time-drifting adaptation using trajectory mixture hidden Markov models. In *Proc. ICASSP '96*, pages 709 – 712, Atlanta, GA, May 1996.

[108] R. A. Sukkar. Rejection for connected digit recognition based on GPD segmental discrimination. In *Proc. ICASSP '94*, pages I–393 – I–396, Adelaide, Australia, April 1994.

[109] R.A. Sukkar and J.G. Wilpon. A two pass classifier for utterance rejection in keyword spotting. In *Proc. ICASSP '93*, pages II–451 – II–454, Minneapolis, MN, April 1993.

[110] E. Thelen, X. Aubert, and P. Beyerlein. Speaker adaptation in the Philips system for large vocabulary continuous speech recognition. In *Proc. ICASSP '97*, pages 1035 – 1038, Munich, Germany, April 1997.

[111] D. Toussaint. Introduction to algorithms for Monte Carlo simulations and their application to QCD. *Computer Physics Communications*, 56:69–92, 1989.

[112] U. Uebler, M. Schussler, and H. Niemann. Bilingual and dialectal adaptation and retraining. In *Proc. ICSLP '98*, volume 5, pages 1815–1818, Sydney, Australia, November 1998.

[113] V. Valtchev, J.J. Odell, P.C. Woodland, and S.J. Young. MMIE training of large vocabulary recognition systems. *Speech Communication*, 22:303–314, 1997.

[114] J. Vermaak and M. Niranjan. Markov chain Monte Carlo methods for speech enhancement. In *Proc. ICASSP '98*, Seattle, USA, May 1998.

[115] T. Waardenburg, J.A. du Preez, and M.W. Coetzer. The automatic recognition of stop consonants using hidden Markov models. In *Proc. ICASSP '92*, San Francisco, CA, March 1992.

[116] B. Wheatley, K. Kondo, W. Anderson, , and Y. Muthusamy. An evaluation of cross-language adaptation for rapid HMM development in a new language. In *Proc. ICASSP '94*, pages I–237 – I–240, Adelaide, Australia, April 1994.

[117] P. Woodland, J. Odell, V. Valtchev, and S. Young. Large vocabulary continuous speech recognition using HTK. In *Proc. ICASSP '94*, pages II–125 – II–128, Adelaide, Australia, April 1994.

[118] P.C. Woodland, T. Hain, S.E. Johnson, T.R. Niesler, A. Tuerk, and S.J. Young. Experiments in broadcast news transcription. In *Proc. ICASSP '98*, Seattle, USA, May 1998.

[119] D. Yuk and J. Flanagan. Telephone speech recognition using neural networks and Markov models. In *Proc. ICASSP '98*, pages 1872–1875, Seattle, USA, May 1998.

[120] G. Zavaliagkos, R. Schwartz, and J. Makhoul. Batch, incremental and instantaneous adaptation techniques for speech recognition. In *Proc. ICASSP '95*, pages 676 – 679, Detroit, MI, May 1995.

Appendix A

Probability distributions

A.1 The normal distribution

The p.d.f. of a k-dimensional multivariate normal distribution is specified as follows [27]:

$$g(x|\mu,\Sigma) = (2\pi)^{-D/2}|\Sigma|^{-1/2}e^{-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)}. \tag{A.1}$$

A normal distribution with mean μ and covariance matrix Σ is usually written as $\mathcal{N}(\mu,\Sigma)$. If we assume that the individual elements of the random variable x are independent (diagonal covariance matrix), we can write Equation A.1 as follows

$$g(x|\mu,\Sigma) = (2\pi)^{-D/2}\left(\prod_{d=1}^{D}\sigma_d\right)^{-1}e^{-\frac{1}{2}\sum_{d=1}^{D}\sigma_d^{-2}(x_d-\mu_d)^2} \tag{A.2}$$

and

$$log(g(x|\mu,\Sigma)) = -\frac{D}{2}log(2\pi) - \frac{1}{2}log\left(\prod_{d=1}^{D}\sigma_d^2\right) - \frac{1}{2}\sum_{d=1}^{D}\sigma_d^{-2}(x_d-\mu_d)^2 \quad (A.3)$$

A.2 The Wishart distribution

Given a random sample of k-dimensional random vectors $(X_1, X_2, \ldots, X_n)$ from a multivariate normal distribution with zero mean and covariance matrix Σ, the random variable V has a Wishart distribution [27] with n degrees of freedom and parametric matrix Σ when,

$$V = \sum_{i=1}^{n} X_i X_i^T. \quad (A.4)$$

For any $k \times k$, positive definite, symmetric matrix v, we have

$$g(v|n,\Sigma) = c|\Sigma|^{-n/2}|v|^{(n-D-1)/2}e^{-\frac{1}{2}tr(\Sigma^{-1}v)}. \quad (A.5)$$

Here, $tr(\Sigma^{-1}v)$ is the trace of the matrix $\Sigma^{-1}v$. The value c is a normalizing constant which ensures that the integral of $g(v|n,\Sigma)$ is equal to one.

If Σ and v are assumed to be diagonal, then Equation A.5 can be rewritten as follows:

$$g(v|n,\Sigma) = c\left(\prod_{d=1}^{D}\tau_d\right)^{n/2}\left(\prod_{d=1}^{D}v_d\right)^{(n-D-1)/2} e^{-\frac{1}{2}\sum_{d=1}^{D}v_d\tau_d} \quad (A.6)$$

where τ is the precision matrix for the Wishart distribution ($\tau = \Sigma^{-1}$) and

$$log(g(v|n,\Sigma)) = log(c) + \frac{n}{2}\cdot log\left(\prod_{d=1}^{D}\tau_d\right) + \frac{n-D-1}{2}log\left(\prod_{d=1}^{D}v_d\right) - \frac{1}{2}\sum_{d=1}^{D}v_d\tau_d. \quad (A.7)$$

A.3 Dirichlet distribution

Given the random vector $X = (X_1, X_2, \ldots, X_k)^T$ with the following properties: For a point $x = (x_1, x_2, \ldots, x_k)^T$ in $\Re^k$, $x_i > 0; i = 1, \ldots, k$ and $\sum_{i=1}^{k} x_i = 1$, then the random vector X has a Dirichlet distribution [27]:

$$g(x|\alpha) = \frac{\Gamma(\alpha_1 + \alpha_2 + \cdots + \alpha_k)}{\Gamma(\alpha_1)\Gamma(\alpha_2)\cdots\Gamma(\alpha_k)} x_1^{\alpha_1 - 1} x_2^{\alpha_2 - 1} \cdots x_k^{\alpha_k - 1}, \tag{A.8}$$

where $\Gamma(\alpha)$ is the gamma function and α is the parametric vector of the distribution and

$$\begin{aligned} log(g(x|\alpha)) =& log(\Gamma(\alpha_1 + \alpha_2 + \cdots + \alpha_k)) - log(\Gamma(\alpha_1)) - \cdots - log(\Gamma(\alpha_k)) \\ &+ (\alpha_1 - 1)log(x_1) + \cdots + (\alpha_k - 1)log(x_k). \end{aligned} \tag{A.9}$$

A.4 The gamma distribution

A random variable X has a gamma distribution [27] with parameters α and β ($\alpha > 0, \beta > 0$) if X has an absolutely continuous distribution whose p.d.f. is

$$g(x|\alpha, \beta) = \begin{cases} \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha - 1} e^{-\beta x} & \text{for } x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{A.10}$$

where $\Gamma(\alpha)$ is the gamma function, which is defined as

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t} dt. \tag{A.11}$$

If a random variable X has a gamma distribution as given in (A.10), then

$$E(X) = \frac{\alpha}{\beta} \tag{A.12}$$

$$Var(X) = \frac{\alpha}{\beta^2}. \tag{A.13}$$

If the mean μ and variance σ^2 of a gamma distribution are known, the distribution parameters α and β can easily be obtained as follows:

$$\alpha = \frac{\mu^2}{\sigma^2} \tag{A.14}$$

$$\beta = \frac{\mu}{\sigma^2}. \tag{A.15}$$

In this thesis, a gamma distribution with parameters α and β has been referred to as $\mathcal{G}(\alpha, \beta)$.

A.5 Conjugate families of distributions

If the prior distribution of θ belongs to a conjugate family of distributions [27], then for any sample size n and any values of the observations in the sample, the posterior distribution of θ must also belong to the same family. A family of distributions with this property is said to be closed under sampling.